



# Teacher Book

when

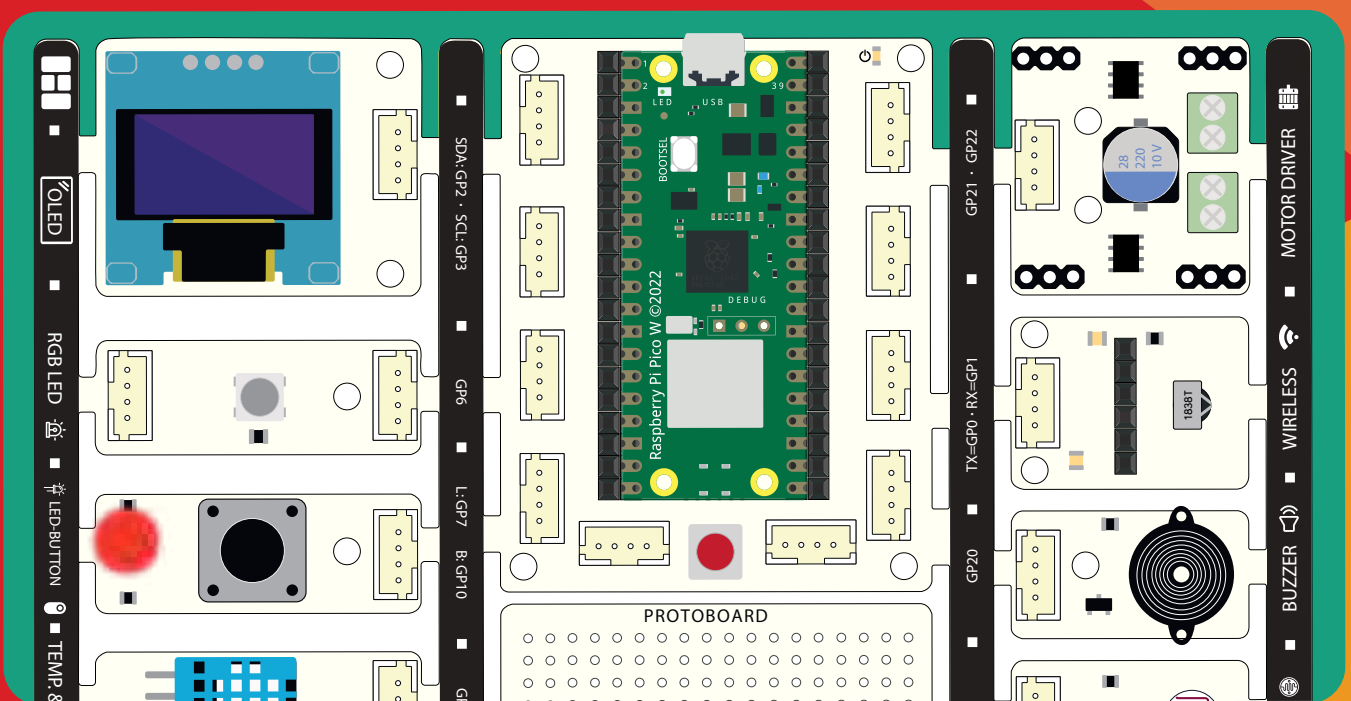
**PicoBricks**

turn off cable clutter

turn off programming difficulty

forever

Develop your creativity with PicoBricks





Copyright © 2023 Robotistan

All rights reserved. It is strictly forbidden to copy, reproduce, use, publish and distribute the text, photographs and other content in this book, in whole or in part without permission, except for individual use.

**Authors:** Selim Gayretli, Hakan Ataş

**Translation:** Naze Gizem Özer

**Design:** Elanur Tokalak

Powered by



The activities in this book are compatible with all PicoBricks versions



Education GitHub



Education Wiki



# CONTENTS

- 1. Introduction .....08
- 2. Education Plan ..... 09
- 3. How Do We Use PicoBricks in Education? ..... 10
  - 3.1 What is PicoBricks? ..... 10
  - 3.2 Recommended Elements to Use with PicoBricks .....11-12
- 4. Some Materials That Can Be Used in The Course Content .....13

## PART 1

- 5. Introduction .....15
- 6. The Problems PicoBricks Eliminates .....15
- 7. Materials that can be used in the course content ..... 16
  - 7.1 What is Computer? .....17
  - 7.2 Processor .....17
- 8. Storage Units ..... 17
  - 8.1 Short-Term Memory .....17
  - 8.2 Long-Term Memory .....17
- 9. Input Devices .....18
- 10. Output Devices .....18
- 11. What is a Microcontroller? ..... 18
- 12. What is Raspberry Pi Pico? ..... 18
- 13. Let’s Examine PicoBricks .....19
- 14. Activity 1 - Let's Get to Know the PicoBricks Modules .....20
- 15. Activity 2 -Unplugged (Computer-Free Activity) ..... 24
- 16. Let’s Test PicoBricks Modules .....25
  - 16.1 Let's Upload Code to PicoBricks with MicroBlocks .....31
  - 16.2 MicroBlocks Code Blocks of the Activity .....31
- 17. Goals of The Lesson .....33



## Part 2

18. Let's Examine The MicroBlocks Blocks .....	35
19. Basic Operations with PicoBricks .....	36
19.1 Let's Install The PicoBricks Library to MicroBlocks IDE .....	36
19.2 Project: Colorful Notes .....	47
19.3 Let's Save Our Project To The Computer .....	48

## Part 3

20. Let's Define a Variable and Enter the Loop .....	50
20.1 What is a Variable?.....	50
20.1.1 Let's Create Variables on the MicroBlocks Platform .....	52
20.2 What is Loop? .....	52
20.2.1 Let's turn on the red LED as much as the value we set using the Repeat loop..	52
21. Counter Activity .....	57

## Part 4

22. Loops with Turtle .....	62
23. Let's Install Turtle Library into MicroBlocks Editor .....	62
22.1 Let's interpret the turtle project below .....	63
23.2 Some Shapes with Turtle .....	64

## Part 5

24. Unplugged: Let's Play Admiral Sunk .....	70
24.1 Admiral Sunk Worksheet .....	73
25. Coordinate Points on the OLED Display .....	74
26. PicoBricks Text Scrolling From Top To Bottom .....	75
27. The Project .....	77

## Part 6

28. Introduction to Conditional Structure .....	80
29. IF-ELSE .....	80
30. Unplugged: Activity .....	81
31. Score Activity .....	82
32. Extra Project: Two-Player Score Activity .....	85
32.1 Project Algorithm .....	85
32.2 MicroBlocks Code of The Project .....	85
32.3 Rock-Paper-Scissors MicroBlocks Project .....	87

## Part7

33. For Loop .....	93
33.1 The Counter with For Loop (1-10) .....	94
33.2 The Counter with For Loop (10-1 ) .....	95
33.3 Counter Incrementing by 2 with For Loop (2-10) .....	95
33.4 Counters decreasing by 2 by 2 with For Loop .....	96
34. The Project of 7th Part: Show Your Reaction .....	96

## Bonus Project: The Number Guessing Game Details

35. Project Algorithm .....	104
36. Project Pin Diagram (PicoBricks v1.0) .....	105
37. Project Images (PicoBricks v1.0) .....	105
38. Project Images (PicoBricks v1.2) .....	105
39. MicroBlocks Code of The Project .....	106

## Part 8

40. Lists .....	110
40.1 Let's Define a List .....	110
40.2 Calling Element From The List .....	111
40.3 Let's Find The Length of The List .....	112
40.4 Add Elements to The List .....	112
40.5 Let's Change an Element in The List .....	113
40.6 Deleting an Element From The List .....	113
40.7 Let's Find an Element in The List .....	114
40.8 Let's Call As Many Elements As We Have Specified Value Range .....	114
41. The Project of The 8th Part: First 100 Digits of Pi Number .....	115

## Part 9

42. Define - Creating Your Own Block .....	119
59.1 The Difference Between Command and Reporter .....	120
59.2 The Shapes with Define .....	121
43. Double The Number with Define (Reporter) .....	124

## STEM Project 1 - Action Reaction

## STEM Project 2 - Dominate The Rhythm

## STEM Project 3 - Dinosaur Game

## STEM Project 4 - Know Your Color

## Fascicle and Worksheet

## Introduction

These days, programming education is gaining importance rapidly with the development of new technology. Educational content covering almost all teaching environments including primary school, secondary school, high school, and university continue to be developed rapidly. This education content should constantly update itself in parallel with the developing technology.

Like every educational process, physical programming education brings a few difficulties. These difficulties may vary with the tools used in physical programming. The main problems experienced in physical programming are; it is divided into two wirings and programming-related problems. The main wiring problems are soldering, cable confusion due to the excess of pin connection points, etc. are problems. The main problems experienced in the field of programming are punctuation marks(Syntax), and memorization problems due to the diversity of programming languages, in comprehensive projects. Difficulty in reading code due to the excess of written codes, etc. faced with many problems. These new tools developed for physical programming education to be used in teaching environments aim to eliminate these difficulties. While eliminating these difficulties, the main goal is not to reduce the quality of education.

In this education book we prepared as the PicoBricks team, we aimed to provide physical programming education using PicoBricks development board. PicoBricks is a development board designed to eliminate the problems experienced in physical programming. Thanks to its modular structure, the cabling and the programming flexibility provided by the microcontroller card Raspberry Pi Pico it contains, eliminates the problems experienced in programming.

For more detailed information about PicoBricks, you can visit [picobricks.com](https://picobricks.com) Website.



## ➤ Education Plan

PicoBricks education plan is designed as 9 chapters + 4 STEM activities. In the education process, the PicoBricks development board and MicroBlocks IDE programming language are used.

The details of the PicoBricks education book chapters are as follows;

PicoBricks Introduction? for loop Bonus Project - Number guessing game

Unplugged- Akinator event Chapter 8 lists Chapter 9 Define functions with STEM

Projects 1. STEM Project Action-Reaction 2.STEM Project Show Your Reaction 3. STEM

Project Dinosaur Game 4.STEM Project Know Your Color writers

<ul style="list-style-type: none"> <li>■ <b>Part 1</b></li> <li>• Unplugged Activities</li> <li>• Introduction to PicoBricks</li> </ul>	<ul style="list-style-type: none"> <li>■ <b>Part 8</b></li> <li>• Lists</li> </ul>
<ul style="list-style-type: none"> <li>■ <b>Part 2</b></li> <li>• Introduction to MicroBlocks</li> <li>• Basic Operations with PicoBricks</li> </ul>	<ul style="list-style-type: none"> <li>■ <b>Part 9</b></li> <li>• Define Functions</li> </ul>
<ul style="list-style-type: none"> <li>■ <b>Part 3</b></li> <li>• Variables</li> <li>• Introduction to Loop</li> </ul>	<ul style="list-style-type: none"> <li>■ <b>STEM Projects</b></li> </ul>
<ul style="list-style-type: none"> <li>■ <b>Part 4</b></li> <li>• Loops</li> <li>• Turtle Library</li> </ul>	<ul style="list-style-type: none"> <li>■ <b>1. STEM Project</b></li> <li>• Action-Reaction</li> </ul>
<ul style="list-style-type: none"> <li>■ <b>Part 5</b></li> <li>• Unplugged - Admiral Sunk Activity</li> <li>• Coordinate Points on the OLED Display</li> </ul>	<ul style="list-style-type: none"> <li>■ <b>2. STEM Project</b></li> <li>• Dominate The Rhythm</li> </ul>
<ul style="list-style-type: none"> <li>■ <b>Part 6</b></li> <li>• Introduction to Conditional Statements(if-else)</li> <li>• Unplugged - Rock Paper Scissors</li> <li>• Rock Paper Scissors with PicoBricks</li> <li>• Score Activities</li> </ul>	<ul style="list-style-type: none"> <li>■ <b>3. STEM Project</b></li> <li>• Dinosaur Game</li> </ul>
<ul style="list-style-type: none"> <li>■ <b>Part 7</b></li> <li>• For Loop</li> <li>• Bonus Project - Number Guessing Game</li> <li>• Unplugged - Akinator Activity</li> </ul>	<ul style="list-style-type: none"> <li>■ <b>4. STEM Project</b></li> <li>• Know Your Color</li> </ul>

## ➤ How Do We Use PicoBricks in Education?



### ■ What is PicoBricks?

PicoBricks is a project development board that aims to eliminate the difficulties experienced in physical programming and to spend the time lost due to these difficulties to come up with more creative projects.

Thanks to its modular structure, Picobricks removes difficulties experienced in physical programming like soldering, cable confusion, incorrect connection of pinpoints, etc. At the same time, the microcontroller card Raspberry Pi Pico is easy to code and it supports many coding platforms, so it eliminates the difficulty experienced in the programming phase.

PicoBricks supports both block-based and text-based programming tools. Thanks to MicroBlocks IDE, we can code our projects in a short time by making block-based coding. It removes many difficulties. In this way, it is a very effective method for developing the algorithm skills required for programming education in young age groups or people who are new to programming. While developing projects with PicoBricks, we can create challenging projects by simply dragging a few code blocks to our project page by using the MicroBlocks program. In addition, PicoBricks supports Arduino IDE C and Thonny IDE MicroPython programming language. Arduino IDE and Thonny IDE programming languages are the most widely used programming tools among text-based programming languages in physical programming education. Since Thonny IDE MicroPython language is supported, PicoBricks eliminates punctuation (Syntax) errors frequently encountered in text-based programming languages.

Considering all these features, PicoBricks aims to increase the quality of education and bring creativity to the fore in physical programming education by eliminating the problems experienced in physical programming education.

## ■ Recommended Elements to Use with PicoBricks

- **Female-Male Jumper Cable:** It provides the connection between Raspberry Pi Pico and sensors.



- **Servo Motor:** It is used to provide movement in mechanical projects that require angular movement.



- **DC Motor:** Provides rotational movement in mechanical projects that require rotational movement.



- **Mini Breadboard:** It is used as an intermediate connection element in projects that require more than one connection.



- **HC-SR04 Ultrasonic Distance Sensor:** It is a circuit element for detecting distance in projects that require distance measurement.



- **RFID NFC Module, Card ve Keychain Kit:** It provides control of circuit elements by using RFID and NFC technology.



- **Bluetooth Module:** It is a circuit element that enables the circuit using Bluetooth technology to communicate with another device using Bluetooth technology.



- **Soil Moisture Sensor:** It is a circuit element that allows measuring the amount of moisture in the soil.



- **ESP8266 Wi-Fi Sensor:** It enables the circuit established in projects that require wireless connection to communicate with other devices via Wi-Fi.



- **Sound Sensor:** It is the circuit element that detects the sound level in the environment.





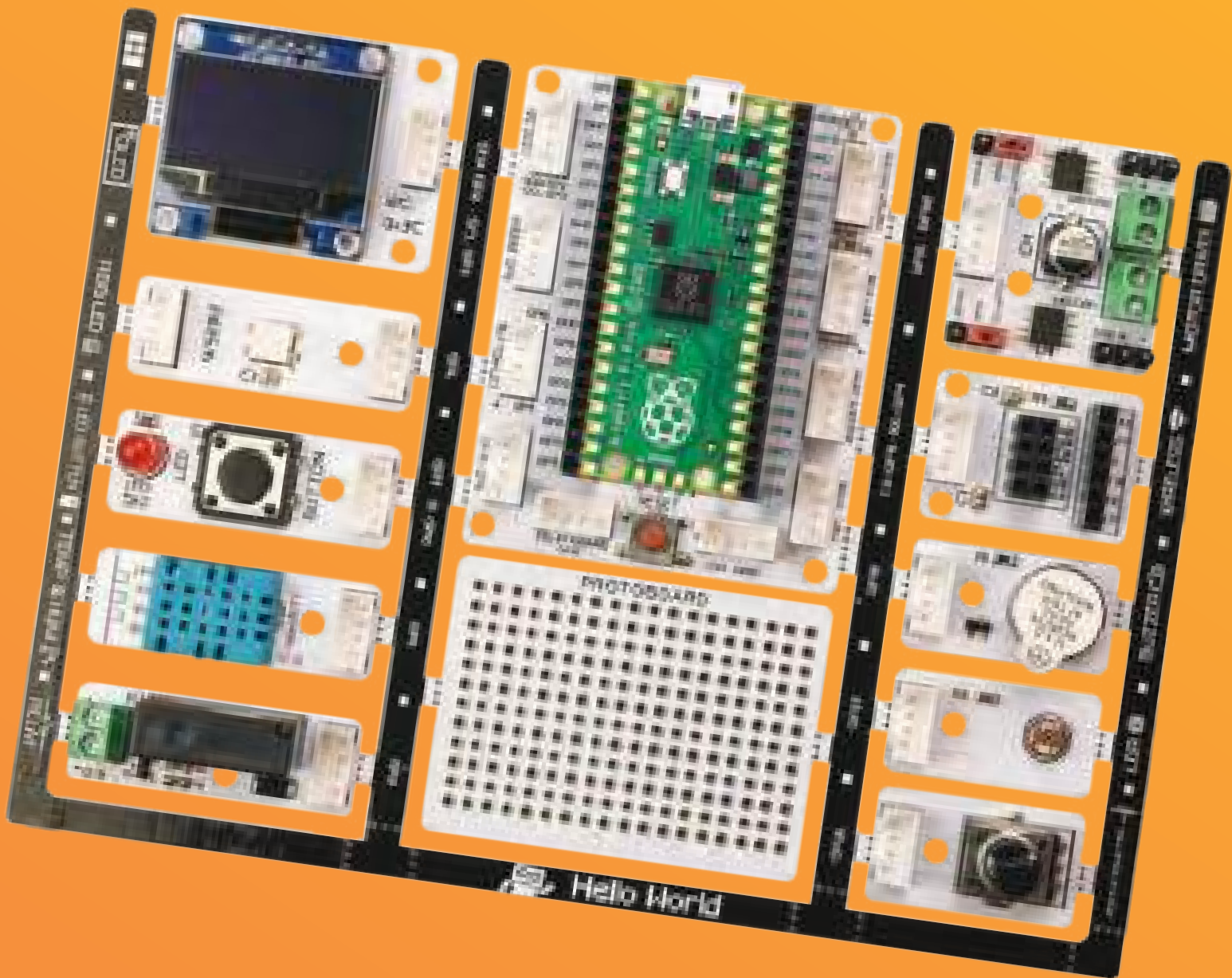
- **PIR Sensor:** It is the circuit element that enables to detection of movement in the environment.



### Some Materials That Can Be Used in The Course Content:

- PicoBricks Box
- Craft Paper
- Waste Cardboard
- Scissors
- Tape
- Color Pencil etc.

# PART 1



## GET TO KNOW PICOBLOCKS EVENTS

In this chapter,

- The student recognizes modules on PicoBricks,
- Understands the working logic of modules on PicoBricks,
- Distinguish between input-output modules on PicoBricks,
- Comprehends the relationship between computers and microprocessors.

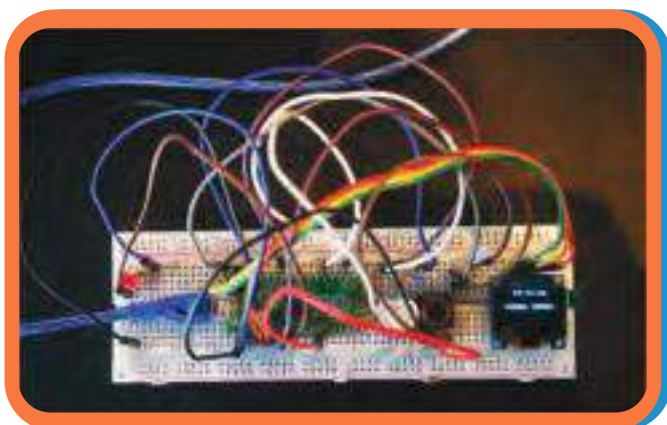
### INTRODUCTION:

Today, coding education is prepared to appeal to students at all levels, from primary school to university level students. PicoBricks is a robotic coding tool that can be easily used in environments that provide robotic coding training for all levels, thanks to its modular structure and flexibility it offers with coding platforms.



## The Problems PicoBricks Eliminates

### No cable clutter



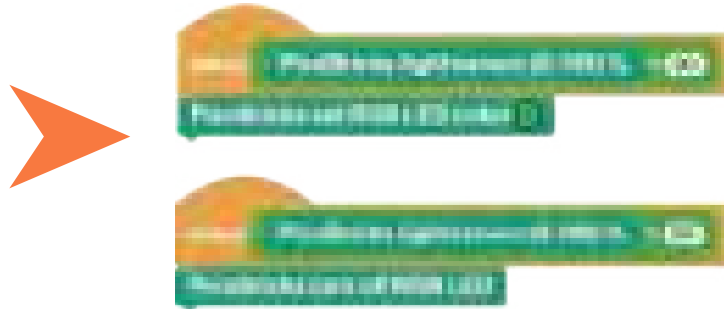
With PicoBricks, you can build your circuits without cable clutter. Similarly, you can get rid of code confusion by coding with the MicroBlocks program. With these feature, we eliminate many problems that we may encounter when starting an ordinary robotic coding education lesson, thanks to the modular structure of the PicoBricks. After entering the robotic coding training,

we have the opportunity to do more projects by breaking the modules on the PicoBricks from the necessary places and connecting the modules to the Raspberry Pi Pico with groove cables.

## Ease of Programming

We want to show you how easy to start coding with PicoBricks and MicroBlocks by sharing an example. We are going to make a project that will make PicoBricks automatically light up in the dark. For the project, we will use the light sensor and RGB LED on PicoBricks. You can find the code blocks of the project below.

Written  
in  
MicroBlocks



Written in  
MicroPython  
with Thonny IDE



In this way, we can teach the logic of coding languages with the text-based Thonny IDE platform after giving a quick start with MicroBlocks to users who have just started coding.

### Materials that can be used in the course content:

- Box of PicoBricks
- Craft Paper
- Waste Cardboard Box
- Scissors
- Duct Tape
- Color Pencil etc.

### Some Advice For The Teacher

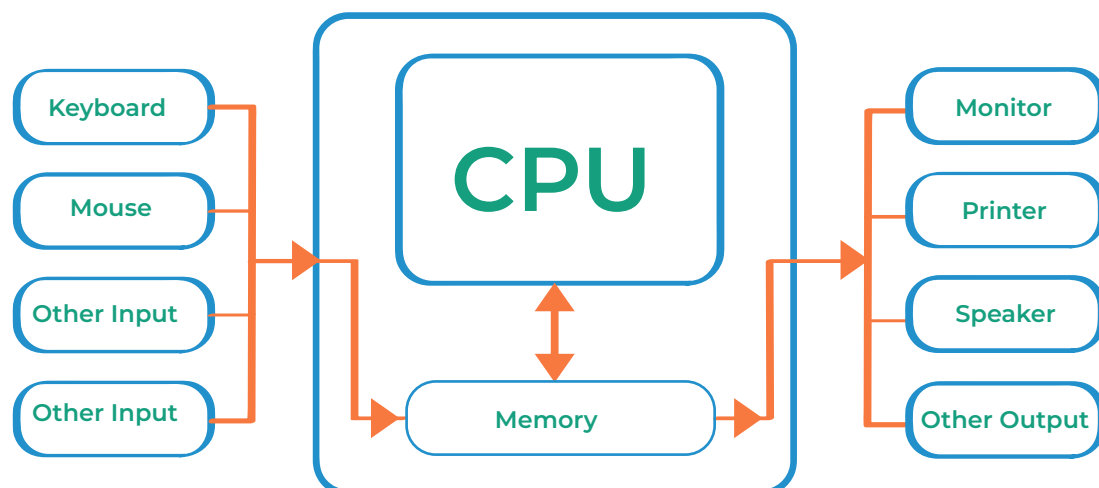
The following information about the lesson plan should be given to the students briefly;

- Get to know PicoBricks activities
- CS unplugged activities
- Let's get to know PicoBricks coding platforms
- Let's start to develop a project with PicoBricks

## What is Computer?

There are 4 main components that make up the computer. These components are;

- Processor
- Storage Units
- Input Devices
- Output Devices




1. **Processor:** Small chips on the motherboard are called processors. We can think of the processor as the brain of the computer. The faster the processor, the faster your computer.


2. **Storage Units:** Computers store information using storage units. Storage units are divided into short-term and long-term memory.

a) **Short-Term Memory:** RAM (Random Access Memory) is the place where information transmitted to the computer is temporarily stored.

b) **Long-Term Memory:** LTM (Long-Term Memory) is a memory unit where information is permanently stored, not deleted.



3.  **Input Devices:** These are devices that allow the computer to receive information from outside. For example, our ears are one of the input organs of the body, and they send the sounds we hear to our brains. The input elements of the computer are the keyboard, mouse, microphone, touch screen, etc. devices that allow us to send the information received from outside to the computer's processor.

4.  **Output Devices:** They are devices through which the computer transmits information. For example, our mouth is one of our organs, we transfer information to the outside with the help of sound by speaking. Computer output devices Monitor, speakers, etc. are devices. The monitor creates the visual outputs and the speaker creates the audio outputs.

## What is a Microcontroller?

Programmable cards that contain a tiny computer are called microcontrollers. Projects can be developed by using input and output sensors together with microcontrollers.

## What is Raspberry Pi Pico?

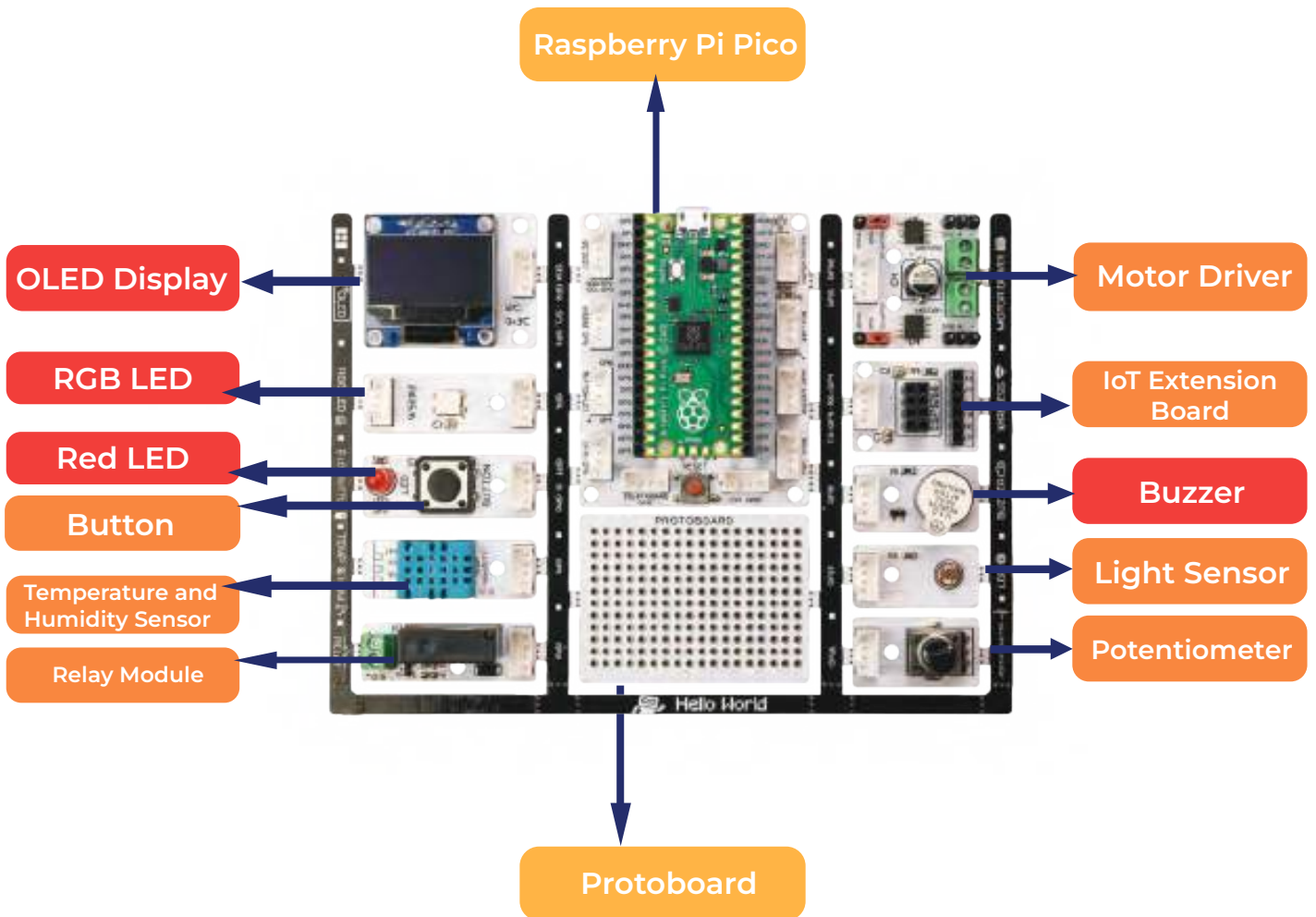
It is the microcontroller board used on Raspberry Pi Pico PicoBricks.

A decorative graphic featuring several question marks of various sizes and colors (orange, light orange) scattered around a central text box. One large orange question mark is prominent on the right side.

### Did you know that Raspberry Pi is a small computer?

Raspberry Pi is a single board computer (Single Board Computer) developed by the Raspberry Pi Foundation in The United Kingdom and made available in 2012. Raspberry Pi Pico is a microcontroller board produced by the same company in smaller sizes.

# Let's Examine PicoBricks

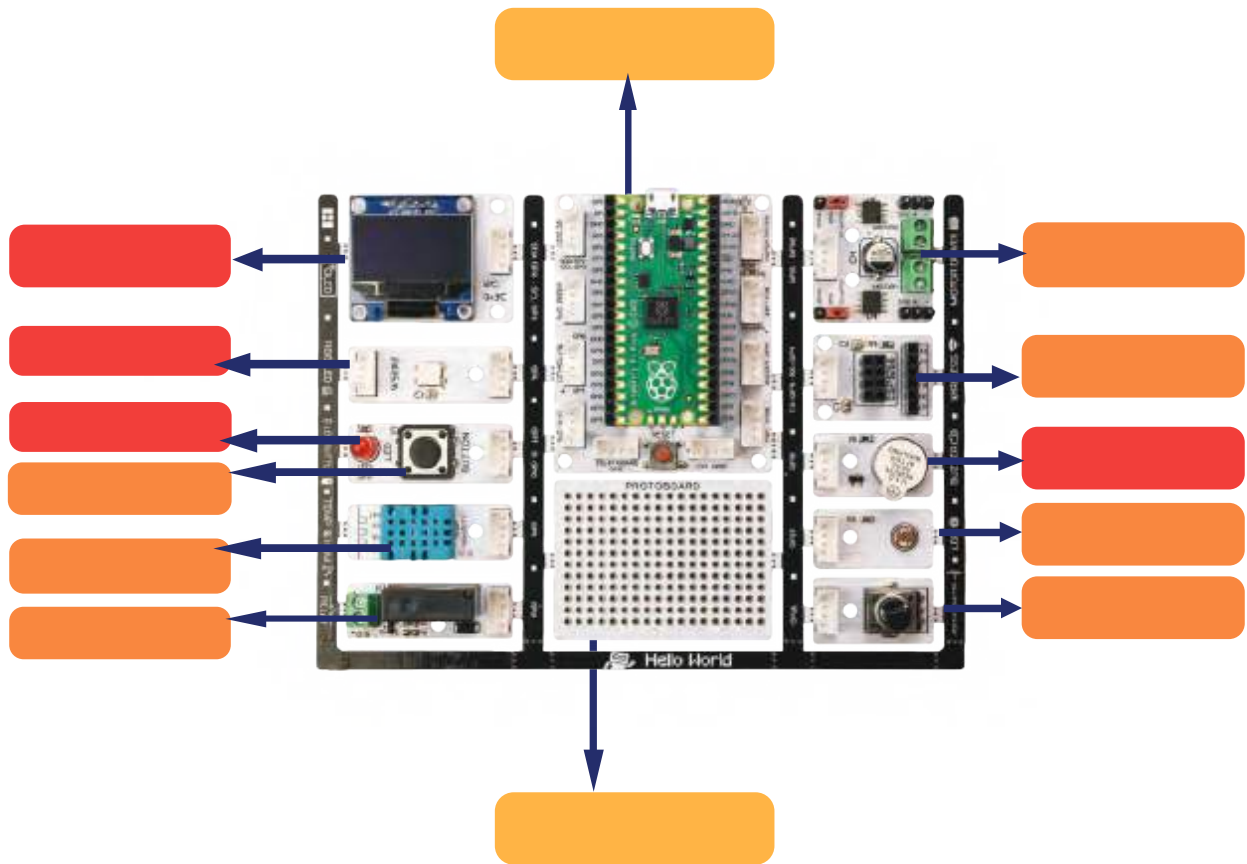


## Some Advice For The Teacher

PicoBricks can be introduced by answering the following questions together with the students. The information learned in this section will be reinforced with the activities.

- How many sensors are on PicoBricks?
- Which of the sensors on PicoBricks are input sensors?
- From which sensor(s) visual output is obtained on PicoBricks?
- Where is the microcontroller board on the PicoBricks?
- What does the protoboard on PicoBricks do?

## Activity 1 - Let's Get to Know the PicoBricks Modules



Before starting the activity, the teacher introduces the sensors on the PicoBricks to the students. The teacher also talks about the input-output devices.

**OLED Display:** It is the module where we can get textual, visual or animation type outputs.

**RGB (Red-Green-Blue) LED:** It is a circuit element that we can output many colors by mixing Red-Green-Blue colors in certain proportions.

**LED & Button:** The red LED module is the circuit element that allows us to output only red light. A button is an input element that we understand whether it is pressed or not.

**DHT Temperature and Humidity Sensor:** It is a module that measures the temperature and humidity of the environment.

**Relay Module:** It is a circuit element that works when current flows through it. You can think of it like a switch that turns on your lamp.

**Raspberry Pi Pico:** It is the microcontroller board that communicates with the sensors in the projects we developed with our PicoBricks board. It also allows us to connect other sensors to the circuit with the pin connection points it has on it.



**Protoboard:** It is the module that enables any sensor or conductive material not on the card to communicate with the card through the conductive paths and jumper cables on the PicoBricks card.

**Motor Driver:** It is the circuit element that adjusts the speed and frequency of the circuit elements like the motor etc.

**IoT Extension Board:** It is the module that provides internet access by connecting to the WiFi network in the environment. Thanks to this module, we can communicate with our projects remotely.

**Buzzer:** It is the module that provides sound.

**Light Sensor:** It is the module that changes the resistance value according to the amount of light falling on it and thus detects the amount of light in the environment.

**Potentiometer:** It is a resistor whose value can change as you turn the knob.



**Do you know who first used the sensors and for what purposes?**

Sensors were first used by Steinel in 1987 for illumination purposes.

**↓ Input Elements:** The elements that transfer the values received from the outside to the computer in the circuit they are connected to are called input elements. For example, DHT temperature and humidity sensor is an input element. Because it transfers the heat and humidity values taken from outside to the computer environment.

**↑ Output Elements:** The elements that output the information received from outside with the help of the circuit visually or audibly are called output elements. For example, the OLED screen is an output element. It gives a visual output of the information it receives with the help of different sensors from the outside.

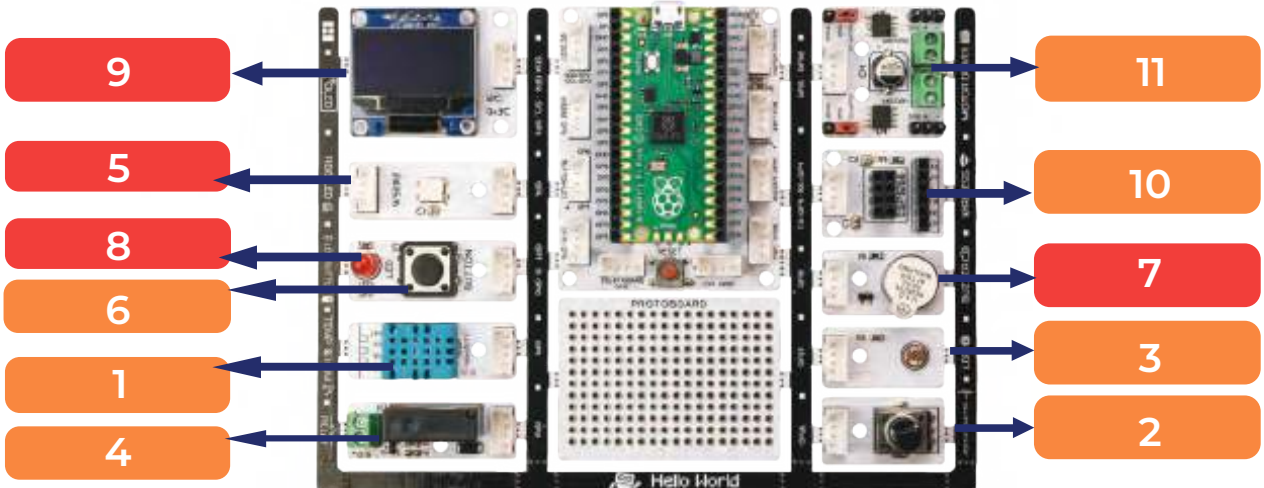
**After the above information is transferred to the students, the activity is introduced to the students.**

The activity design given above can be drawn on the classroom board, reflected with a projector, printed out, and distributed to each student, or positioned at a point where each student can see it easily. Afterward, the teacher asks the students to write the module, which is the answer to the question, in the correct box by conveying the following questions about the modules. These questions are read one by one during the lesson and students are expected to place the answer in the correct box.

## Sample Questions:

- What is the circuit element that allows us to output an image, text, etc. on PicoBricks? Write it in the relevant box on PicoBricks. (OLED Screen)
- What is the output element that provides the color base by mixing red, green, and blue colors at certain values? Write it in the relevant box on PicoBricks. (RGB LED)
- What is the circuit element that allows us to output only red light? Write it in the relevant box on PicoBricks. (Red LED)
- What is the circuit element that causes a process to start and end by applying pressure on it? Write it in the relevant box on PicoBricks. (Button)
- What is the circuit element that allows to measure the amount of heat and humidity in the environment? Write it in the relevant box on PicoBricks. (DHT Sensor)
- What is the circuit element that cuts the current flowing through the circuit or allows the current to pass? Write it in the relevant box on PicoBricks. (Relay Module)
- What is the module that allows the coding of the modules on the PicoBricks with the pinpoints on it, which allows the connection of different modules that are not on the PicoBricks? Write it in the relevant box on PicoBricks.(Microprocessor Module)
- What is the circuit element that controls the speed and frequency of the DC motors connected to the circuit? Write it in the relevant box on PicoBricks. (Motor Driver)
- What is the circuit element that allows wireless control of a project prepared with PicoBricks? Write it in the relevant box on PicoBricks. (ESP Module)
- What is the circuit element that creates audio output on PicoBricks? Write it in the relevant box on PicoBricks. (Buzzer)
- What is the circuit element that detects the amount of light in the environment and converts it into a digital value? Write it in the relevant box on PicoBricks. (LDR Sensor)
- What is the circuit element that changes the resistance value applied to the circuit with external physical interventions? Write it in the relevant box on PicoBricks. (Potentiometer)
- What is the module that allows different sensors or conductive materials to be connected to the PicoBricks, thanks to the conductive points it has on the PicoBricks? Write it in the relevant box on PicoBricks. (Protoboard)

Right after this activity, the boxes related to the modules are numbered using the same activity design for a better understanding of the Input and output modules, and the activity design is positioned at a point where every student can see it.



Then, the activity design below is positioned at a point where every student can see it.

## Let's Get to Know the Input and Output Modules



(hint: The largest odd number with different digits)

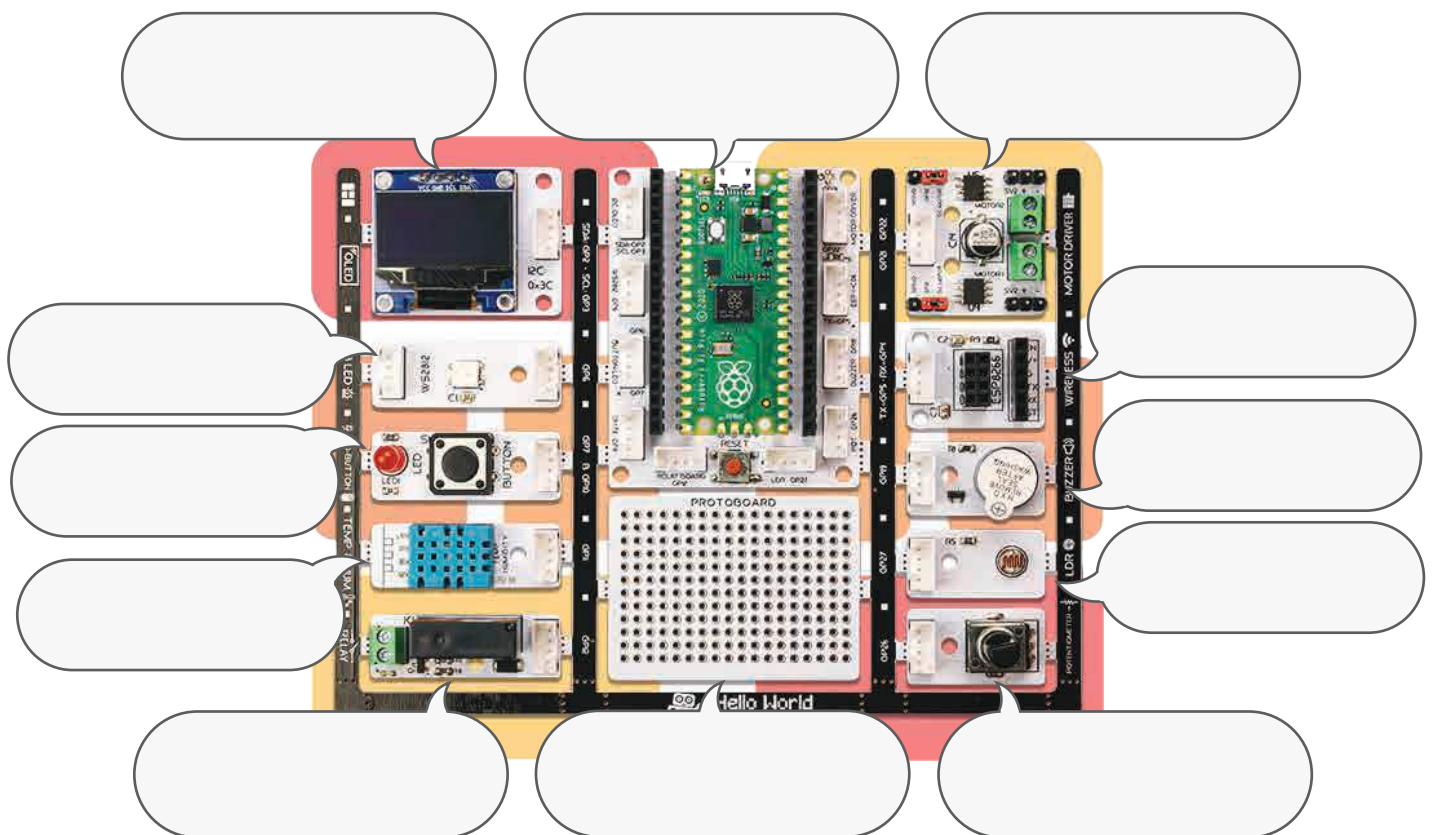
- Module with red light output
- A module that outputs text, images or simple animations
- Module that outputs sound
- Module that gives colored light output by mixing red, green, blue colors.



This activity design can be printed out and distributed to the students, the students can be drawn in their notebooks or the teacher can be positioned on the board where every student can see it.

- 1) The teacher asks the students to correctly place the module numbers in the 1st activity design in the boxes given in the 2nd activity design. This activity can be done on the board with the question-answer technique, and at the same time, students can be expected to do it in their own notebooks.
- 2) Then, the teacher explains the necessary clues to the students in order to write the correct numbers in the correct order in the 4 spaces given in the 2nd activity design;
  - Identify the numbers using the 4 clues under the blanks and the module numbers in the 1st activity design.
  - When the numbers you have determined are brought together, the largest odd number with different digits is obtained.

## Activity 2 (Unplugged Activity)



## Let's Get to Know the Modules - PicoBricks Speech Bubbles Activity

The teacher starts this activity by drawing a speech bubble for each module on the PicoBricks card as in the picture below. The aim of the activity is to write the correct sentences on the speech bubbles that concern the modules. Example sentences should be as follows;



**Buzzer:** If you want to give an audible warning in your project, you should use me.



**OLED Screen:** I provide you to output text or images in your projects.

**LED:** If you want to get a print out with only red light in your projects, you should use me.



**Button:** You can provide starting or finishing events in your projects by putting pressure on me.



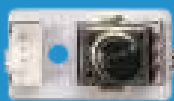
**RGB LED:** I can create the color output you want by mixing red, green and blue colors in certain proportions in your projects.



**DHT Temperature and Humidity Sensor:** If you want to use the temperature and humidity value of the environment in your projects, you should use me.



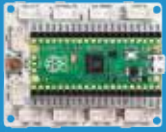
**Relay Module:** In your projects, I enable you to turn the circuit on and off by cutting the electric current or allowing the current to circulate in the circuit.



**Potentiometer:** It is a resistor whose value can change as you turn the knob.



**LDR Sensor:** If you want to detect the amount of light in the environment in your projects, you should use me.



**Raspberry Pi Pico Module:** You use me to code your projects, and you also make the pin connections of the modules and sensors you use from the pins on me. To make an analogy, I am the brain of the PicoBricks.



**Protoboard:** You can connect any conductive material to the circuit with the conductive paths on me and assign a value to this item in the code thanks to me.



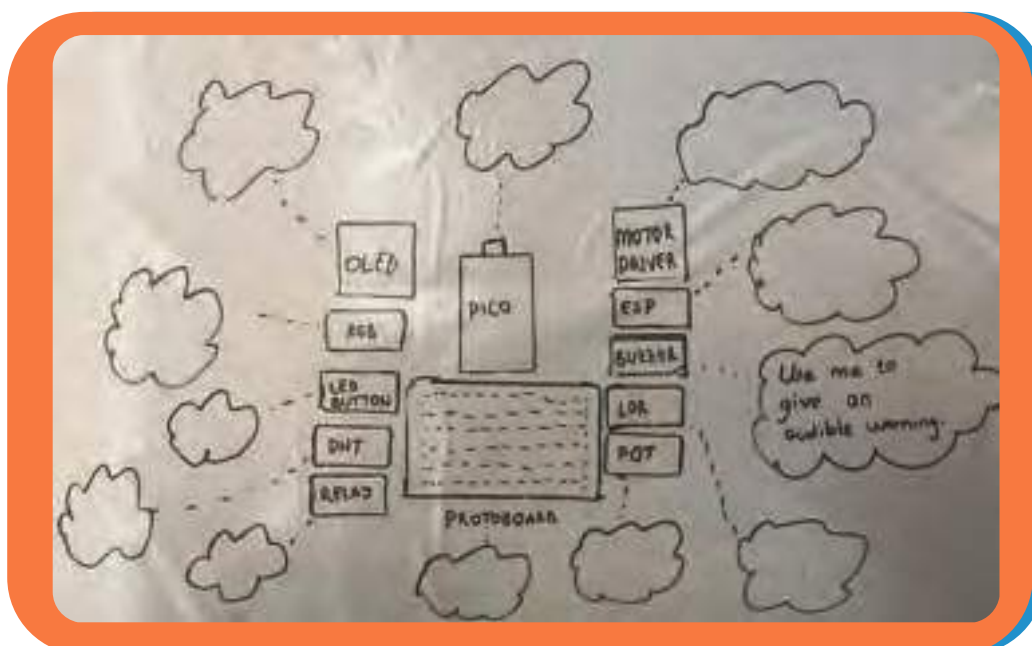
**Motor Driver:** If you want to use DC motor or servo motor in your projects, you must use the circuit me in order to adjust the speed and frequency of the servo motor and DC motor.



**Wireless Module:** You need to use me to communicate via bluetooth or Wi-Fi in your projects.

The activity can be applied to the classroom in different ways by the teacher. A few of these applications are listed below.

- The prepared sample sentences are read in order and the student is asked to write the sentence in the correct speech bubble. Attention should be paid to the participation of each student in the activity.





Each student draws the sample activity design in their notebook (a very detailed drawing should not be expected.) or the sample design is printed out and distributed to each student. Afterward, the teacher reads the sample sentences prepared for everyone to hear in the classroom and waits for the students to write the read sentences in the speech bubble on the correct module.



The main goal of this activity is for the student to get to know the PicoBricks. The student understands the modules on the PicoBricks and the communication between the PicoBricks and these modules and has an idea for their future projects.

#### Achievements:

- The student recognizes PicoBricks.
- The student understands the working logic of modules on PicoBricks.
- The student recognizes sensor technologies.

#### Some Advices for The Teacher

In order to increase the permanence of the information learned at the end of the activity, the activity should be supported with sample questions.

#### Sample Questions:

If you wanted to make a project using the modules on PicoBricks, what kind of project would it be and which modules would you use? Explain.

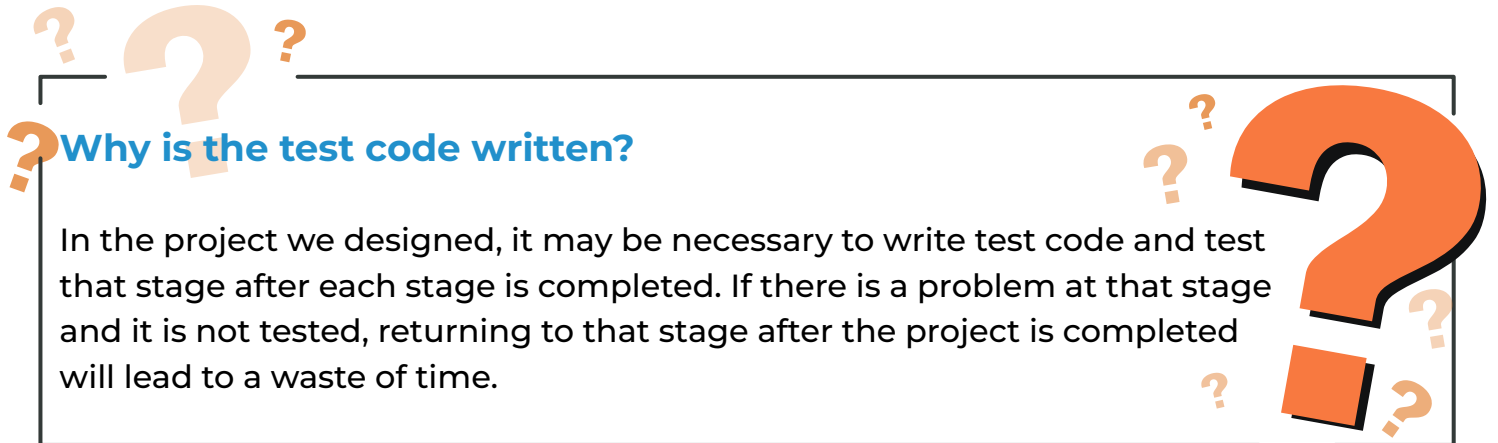
What are the input elements on PicoBricks? Please order.

What are the modules that give light output on PicoBricks? Please order.

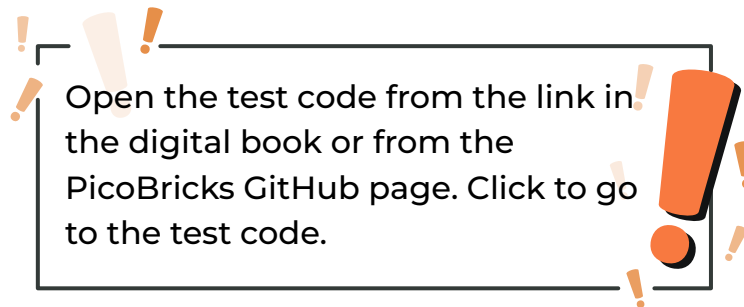
Let's load the test code written with MicroBlocks into PicoBricks and run it to test that the PicoBricks modules work and observe how they work.

**Why is the test code written?**

In the project we designed, it may be necessary to write test code and test that stage after each stage is completed. If there is a problem at that stage and it is not tested, returning to that stage after the project is completed will lead to a waste of time.



Open the test code from the link in the digital book or from the PicoBricks GitHub page. Click to go to the test code.



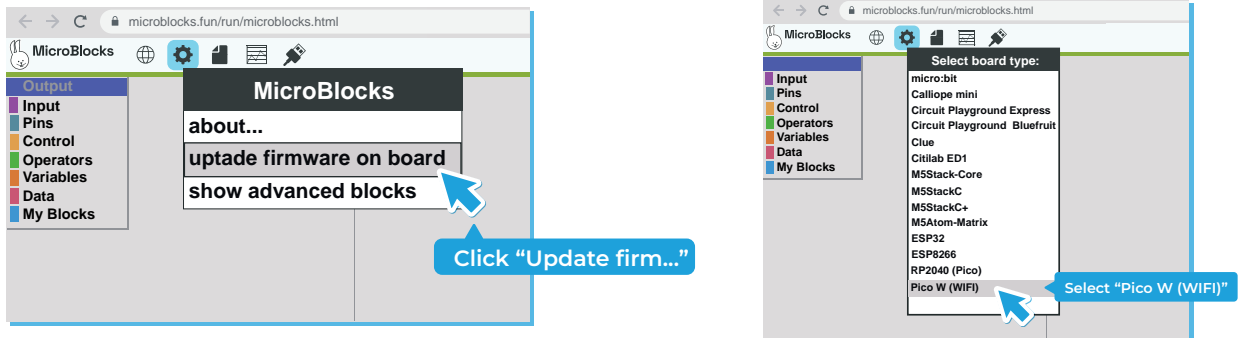
## Bricks with MicroBlockLet's Upload Code to Pico's

1. Let's Upload Code to PicoBricks with MicroBlocks

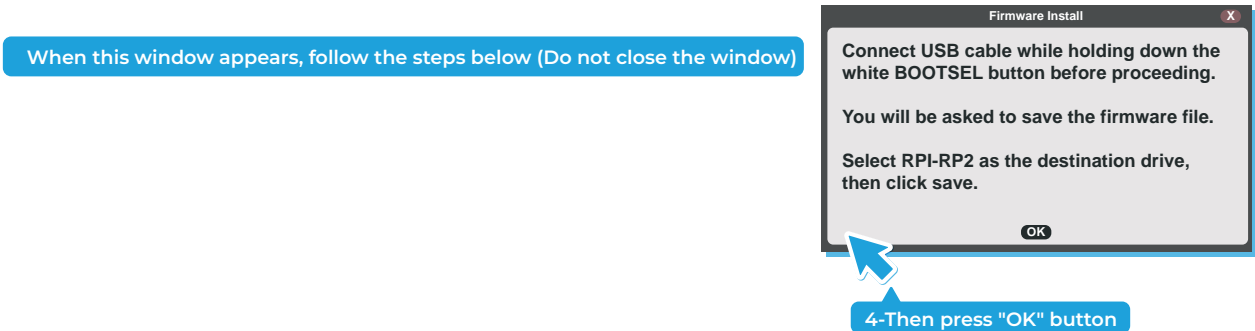




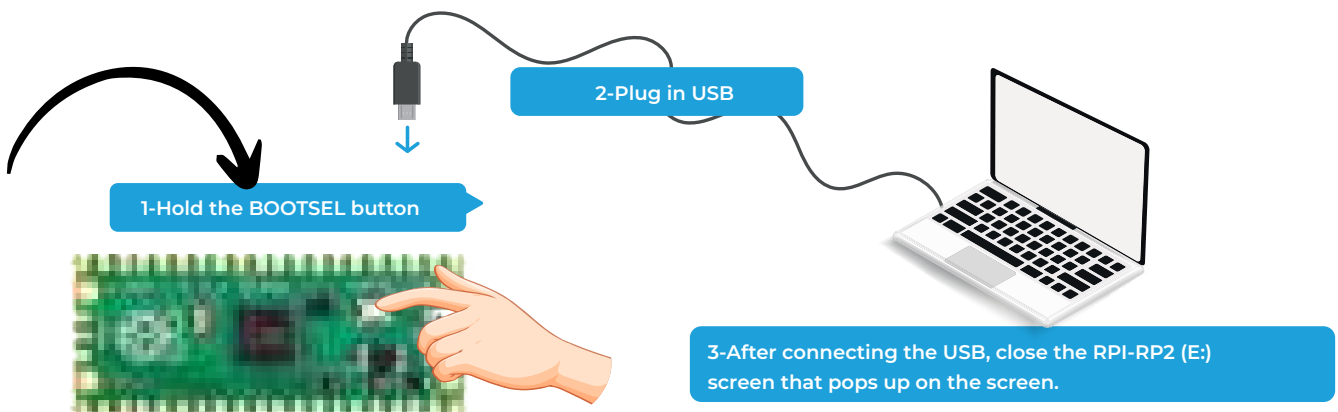
2. Now, let's follow the steps below to connect the PicoBricks to the computer. First, click the settings icon, then click the "update firmware on board" option from the window that opens. From the other window that opens, select the "RP2042 (Pico)" option.



3. Press the "OK" button in the BOOTSEL window that opens. Do not close the save window that opens later.



4. To connect PicoBricks to the online editor, you must connect the card to your computer with the USB cable while holding the white BOOTSEL button on the Raspberry Pi Pico.



6. In the save window opened in step 3, save the "vm\_pico" file in RPI-RP2 (D:).

If the icon is green, it is connected

7. Clicking the Connect button will display the system USB ports where the micro devices are plugged in. In this window, you can connect PicoBricks to MicroBlocks by first selecting the Pico device and then clicking the Connect button. When the connection is successful, a green circle will appear behind the USB icon.

Click the button to run the PicoBricks test code. Click the button to stop the code.

## Activity Image



## MicroBlocks Code Blocks of the Activity



### Activity Details

All modules except Protoboard module on PicoBricks with PicoBricks test code we can test it.

When we press the button, "Hello Picobricks" is written on the OLED screen.

We print the temperature and humidity value using the DHT temperature and humidity module.

We run the red LED module and the RGB LED module and get color outputs.

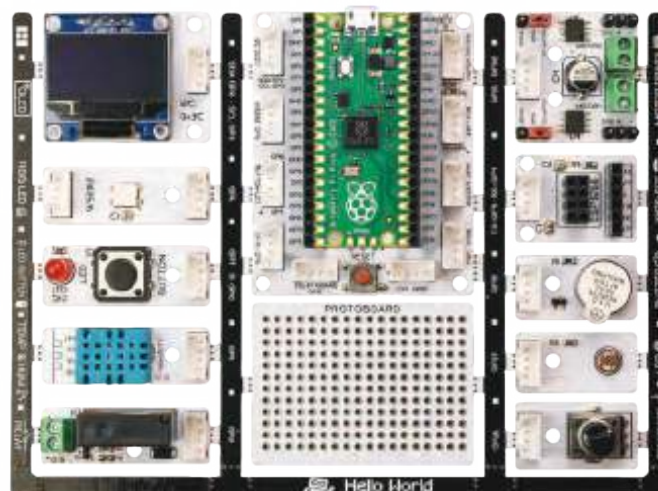
We turn the relay on and off after waiting for 1 second. You can understand whether the relay is on or off by the small LED light on the relay module and the opening and closing sound.

We activate the servo and DC motor points that can be installed on the motor driver module, and if we attach a DC motor and servo motor to the motor driver pin points, we can observe that it works.

In addition, it can be tested whether the light on the sensor is on by attaching the bluetooth sensor to the pin point for the bluetooth sensor on the picobricks ESP module, and if it is, the module is working.

We get sound output from the buzzer.

We cover the LDR sensor module with our hands and turn on the RGB LED.



By changing with the setting of the potentiometer, we can see the change in the potentiometer value on the OLED screen.

## Some Advices For The Teacher

1. Using the explanations above, introduce the students to the sequence of how the test code works.
2. After sharing the Test Code with each student, run the code with the students.
3. Explain the modules on PicoBricks while the code is running and repeat the information learned in the first lesson about their working logic.
4. Check that the motor driver is working by first attaching a yellow (DC) motor to the pin points.
5. Observe that the servo motor is working by attaching a servo motor to the pin points of the motor driver.
6. Draw attention to the difference between the servo motor and the Yellow (DC) motor operating logic and explain this difference to the students.
7. Observe the change of the potentiometer value on the OLED screen by playing with the potentiometer.
8. Show that the RGB LED is working by covering the LDR sensor with your finger and explain the working logic of the LDR sensor.

## Goals of The Lesson:

- It is aimed that the student gets to know PicoBricks.
- It is aimed for the student to know about project development with PicoBricks.
- It is aimed for the student to comprehend the relationship between computer and microprocessor.
- It is aimed that the student will develop hand-arm coordination while designing.
- The steps of the student's computational thinking skills;
  - Pattern matching skills,
  - Algorithmic thinking skills,
  - Decomposing skills
  - The ability of abstraction, are aimed to be gained.
- Identifying the problems around the student, is aimed at approaching the problem by considering the steps of computational thinking skills.

## K-12 International Computer Science Standards

• 1B-AP-08 - 3-5 Compare and refine multiple algorithms for the same task and determine which is the most appropriate.

• 1B-DA-07 - 3-5 Organize and present collected data visually to highlight relationships and support a claim.

• 1B-DA-07 - 3-5 Use data to highlight or propose cause-and-effect relationships, predict outcomes, or communicate an idea

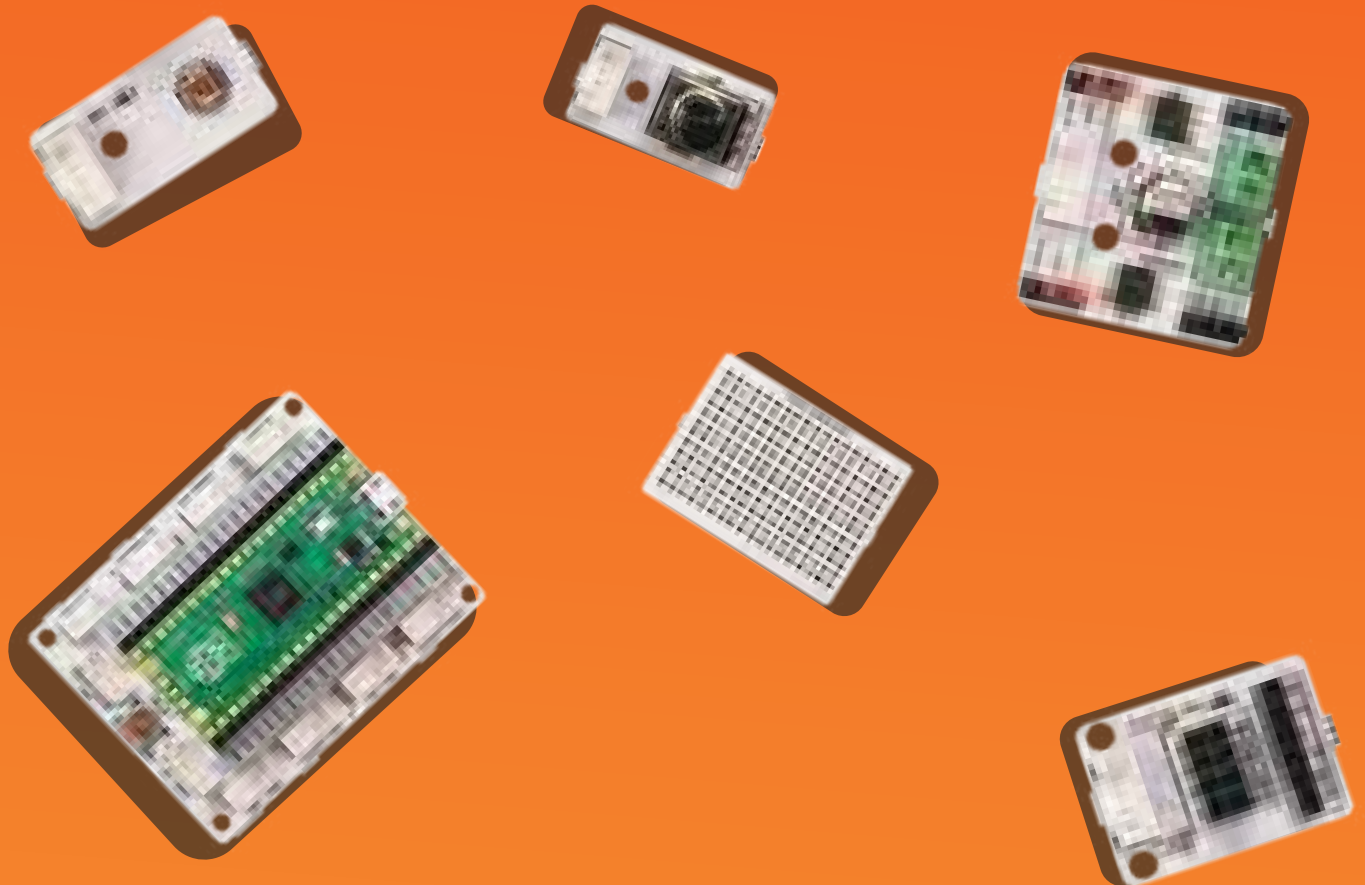
• 1B-AP-08 - 3-5 Compare and refine multiple algorithms for the same task and determine which is the most appropriate.

• 1B-AP-11 - 3-5 Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.

• 1B-AP-13 - 3-5 Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences.



# PART 2



## What We'll Learn in This Part

### The student, in this part;

- Learns the MicroBlocks block structure,
- Can load the PicoBricks Library on the MicroBlocks project page,
- Can upload code to the project page,
- Can delete code from the project page,
- Using the PicoBricks library, it can run PicoBricks modules.

## Let's Examine The MicroBlocks Blocks

**Output Blocks:** Code blocks that manage audio and video outputs.

**Pin Blocks:** Code blocks that we use when we connect to PicoBricks pins.

**Operator Blocks:** Code blocks with mathematical operations.

**Data Blocks:** Code blocks that allow us to use the data we have defined for the project.



**Input Blocks:** Code blocks that manage input devices.

**Control Blocks:** Code blocks where we can control run, end, condition, loop etc.

**Variable Blocks:** Code blocks where defining a variable, assigning a value to a variable etc. are made.

**My Blocks:** Code blocks where we can prepare our own blocks and define tasks.

**Libraries:** The section where we can access the code blocks after loading the libraries into the MicroBlocks IDE editor.

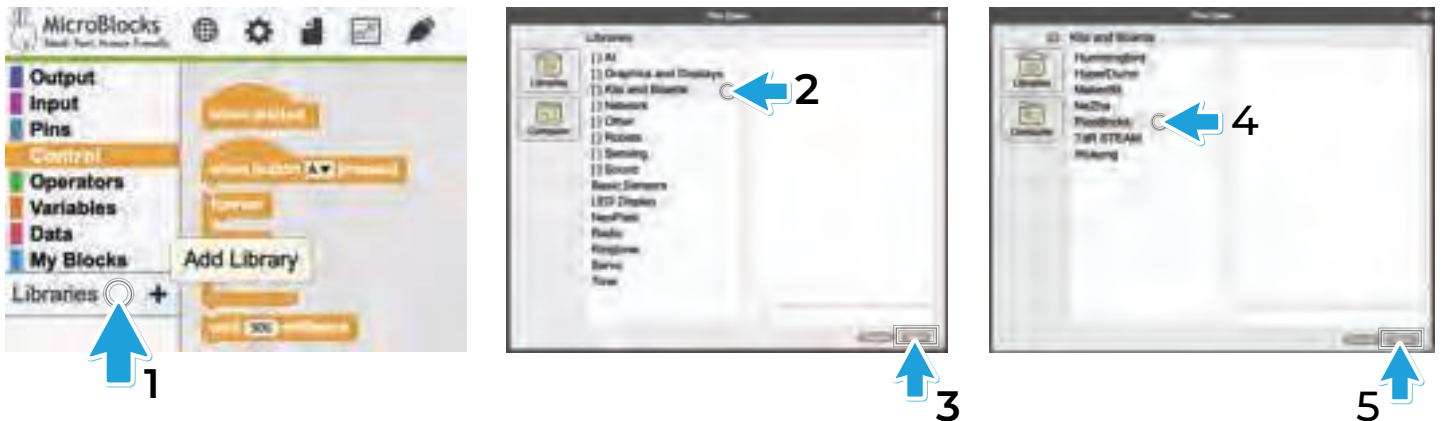


## Basic Operations with PicoBricks

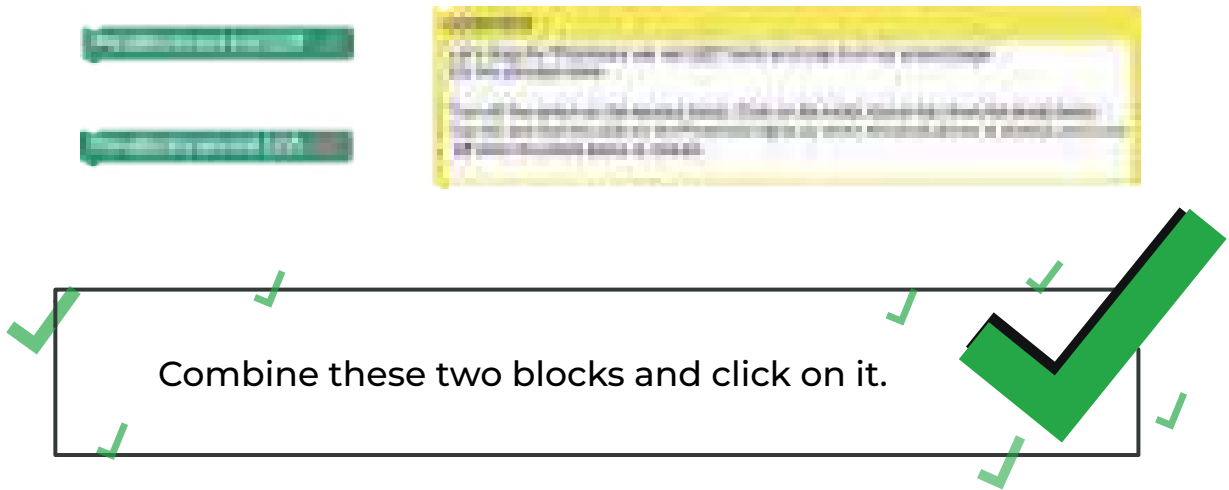
Let's open MicroBlocks online editor and connect PicoBricks (Let's see the icon is green ). Then, load the PicoBricks library into the editor

### Let's Install The PicoBricks Library to MicroBlocks IDE

Let's install the PicoBricks library by following the steps below.



### Let's Turn On and Turn Off The Red LED Module



You will observe that the red LED on the PicoBricks is not lit. In fact, the red LED is on and goes off quickly, but because this process happens very quickly, we cannot see the LED light up.

In MicroBlocks, you can run the relevant block(s) by simply clicking on the code block(s) you want to run.



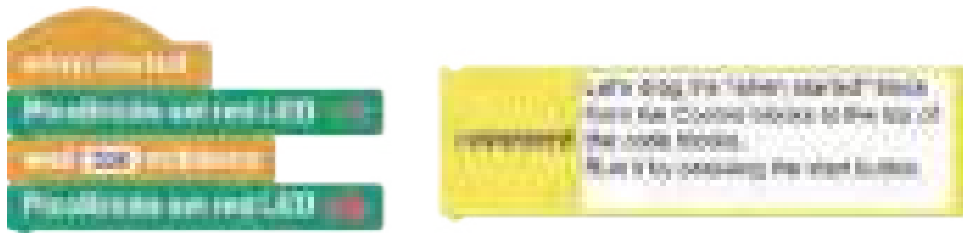
Let's use the "wait" block to wait half a second between blocks.



Click start button in the upper right corner of the MicroBlocks editor.

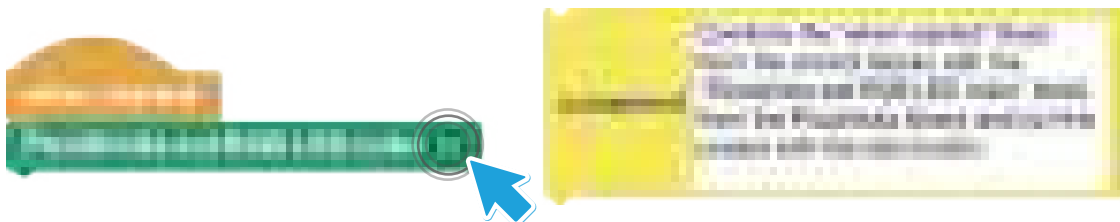
When you click the start button, you will observe that the code does not work. This is because we do not use any star command blocks.

Now, let's connect the blocks using the "when started" block and press the start button.



You will observe that the red LED on the PicoBricks blinks once.

## Let's Use the RGB LED Module



We just flashed the RGB LED module in green. This color comes as the default color when we drag the RGB LED block from the PicoBricks library. To change this color, click on the green dot and select the color you want from the color chart and run the program.



We have learned the working logic of RGB LED before, now let's repeat it briefly. RGB LEDs provide different colors by mixing red, green and blue colors in a certain ratio. We can adjust the color values ourselves by using the PicoBricks color RGB block in the PicoBricks library. (Color value can be maximum 255.)



**Do you know how many different colors you can get with RGB LED?**

We can create  $256 \times 256 \times 256 = 16,777,216$  different colors with RGB LEDs.



Now, let's light up a color that we determined by ourselves (0-255) with a RGB LED module.

**when started**

**PicoBricks set RGB LED color**

PicoBricks color r 250 g 90 b 158 (0-255)

**comment** Drag the Picobricks color RGB block to the colored dot in the right corner of the "Picobricks set RGB LED color" block. Then enter the R(Red) G(Green) B(Blue) values.

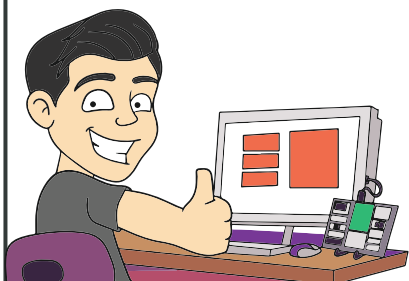


## Now It's Your Turn!

Let's light the RGB LED module in red, green and blue colors at half-second intervals.

## Let's Make Sound Using the Buzzer Module

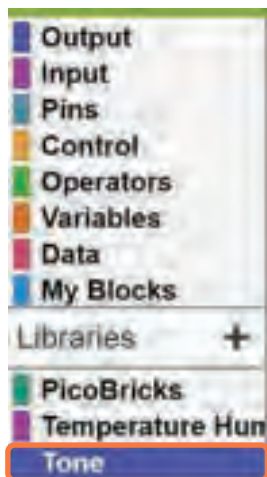
Let's make a simple warning sound with the buzzer.

## Now It's Your Turn!

Make a warning project using the red LED and Buzzer on the PicoBricks. Let's make different tones with the buzzer by using the MicroBlocks Tone library.

Load the Tone library into the MicroBlocks IDE online editor from the Libraries tab.





Drag the play note block from the tone library under the when green flag clicked and press the play button.

Observe the change of the sound coming out of the buzzer by changing the values in the block we dragged from the Play Tone library.



Let's produce three different tones by changing the values in the note block that we dragged from the tone library. Observe the difference between the sounds by using the wait half-a-second block between them.



You can change the values in the blocks as you wish.

Now let's explore the other blocks in the tone library.

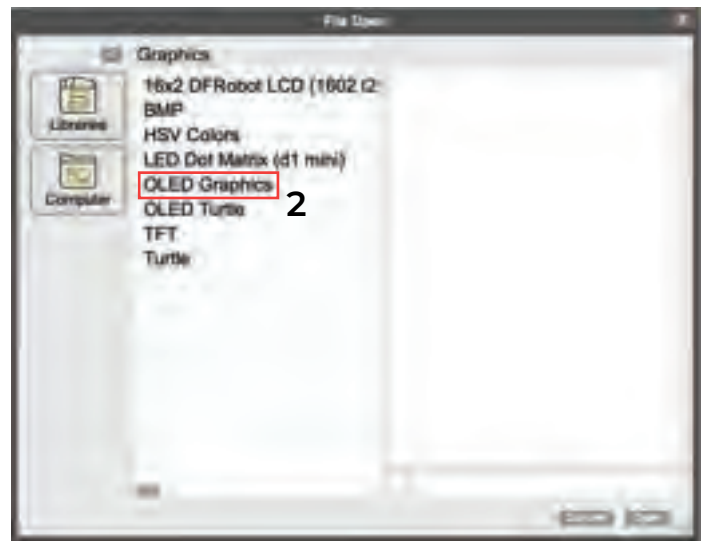
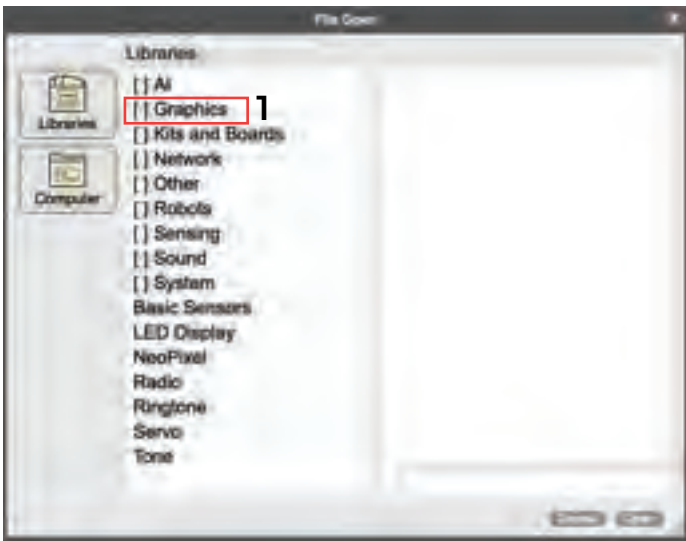


There are three options in the tone library: play, when green flag clicked, and when green flag clicked and press the play button. The play and when green flag clicked blocks will play the sound when the script starts. The when green flag clicked and press the play button block will play the sound when you click the play button.

## Let's Make Sound Using The Buzzer Module

Let's print on the display using the OLED display module

Being able to print on the OLED display is one of the taking output methods. In this section, we will see how to use the OLED display. To use the OLED display in MicroBlocks, we need to load the OLED Graphic library from the libraries into the MicroBlocks IDE editor. Install the library by following the steps below.

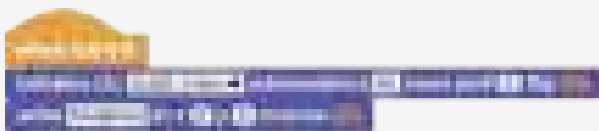


In order to use the OLED screen on PicoBricks, we need to initialize it by using the following block, this block should be used before writing anything on the OLED screen.

**Initialize i2c** `OLED_0.96in` address(hex) `3C` reset pin# `0` flip

"This is the default address for PicoBricks, we should not change it"

Now let's print our name on the OLED display.






Now print your name in the middle of the display using the X and Y coordinates.


## Let's Start Our Project Using The Button

In all the activities we have done so far, we have started our projects using the start button on the MicroBlocks IDE platform. Now, we can start our projects using the button on Picobricks.

In order to start the project, use the button on PicoBricks;

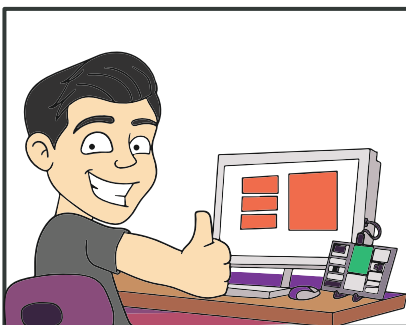
Drag  block from "Controls" blocks to your project page.

Drag  the block from the PicoBricks library into the key mark on  the block.

Now, after pressing the  button, we can run the blocks that we drag under the code block by pressing the button on the PicoBricks thanks to the

 block.

Now, after pressing the button, let's drag the code blocks that run the red LED and Buzzer to our project page.

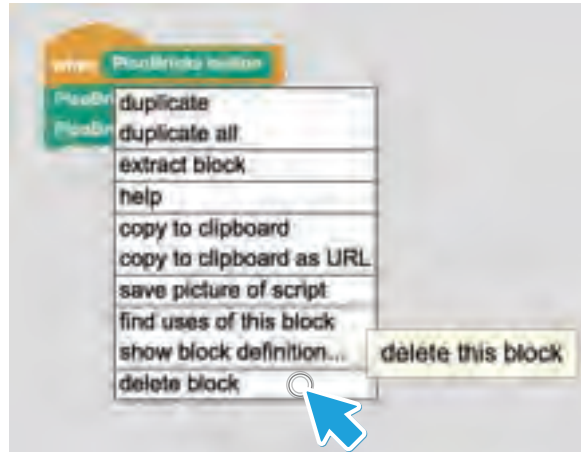


### Now It's Your Turn!

After pressing the button with PicoBricks, drag the code blocks that flash the RGB LED in red, green and blue at half-second intervals, then activate the buzzer and print "PicoBricks" on the OLED display, onto your project page.


## Let's Delete A Code Block From The Project Page

When we want to delete a code block on our project page, we can delete it by right-clicking on that block and clicking the “delete block” button.



We can also delete a code block(s) on our project page by dragging it back into the code blocks area on the left. Let's delete the code blocks on our project page by dragging them to the blocks tab on the left.



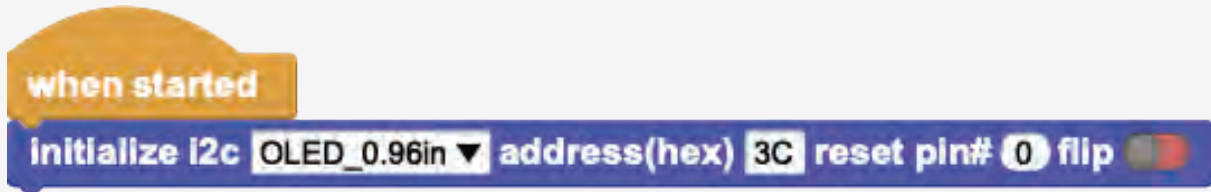
Let's replace the code that runs the buzzer and the red LED at the same time when the button is pressed, which is the last application we made, with the code that runs when the program is started, that is, when the  button on the top right of the project page is pressed. We don't need to open a new project for this, let's delete the top block from the code blocks on our project page and replace it with the "when started" block.



## Let's Use Temperature and Humidity Sensor Module

Now, let's drag the code blocks that print the temperature value of the environment we are into the project page using the temperature and humidity sensor module.

Let's drag the following two code blocks to our project page to start the project and the screen.



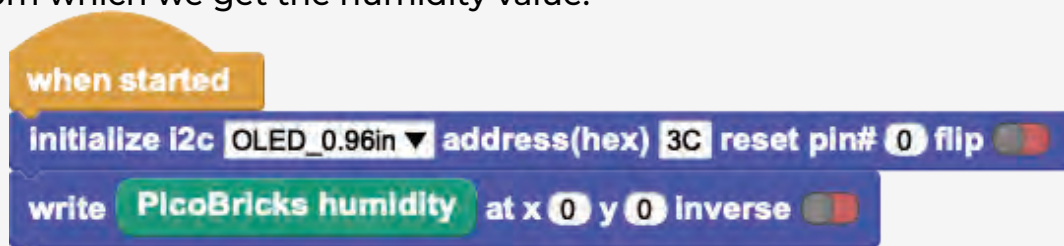
Now, let's drag the code block we use to print a value to the screen from the OLED library to our project page. Then, drag the code block that allows us to get the heat value from the PicoBricks library on this block as shown in the picture below.



After combining this code block with the other two code blocks, we can run our project.




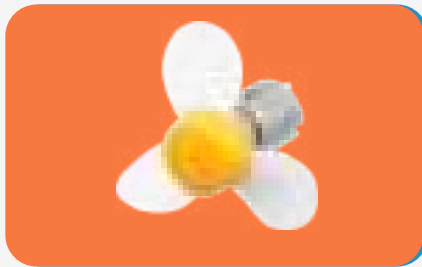
Let's print the humidity value of the environment on the screen using the 'PicoBricks humidity' block in the same way. We can do this by simply deleting the block that prints the temperature value from our project page and dragging the block from which we get the humidity value.





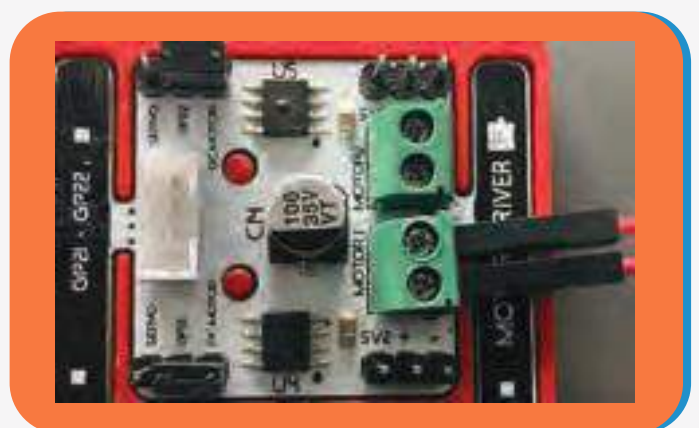
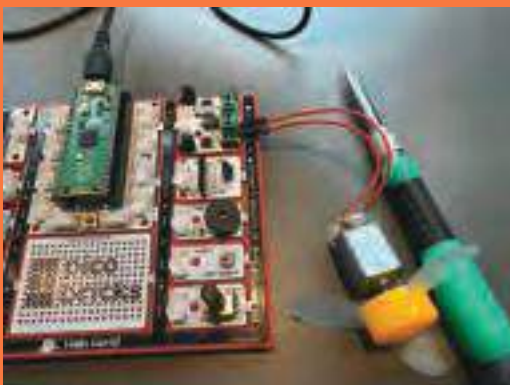
## Let's Run The Propeller Using The Motor Driver Module


Let's run the DC motor included in the set by using  block in the Picobricks library and let's make ourselves a simple push-button propeller.

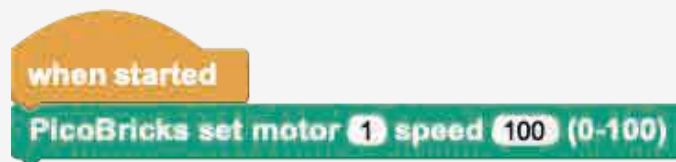
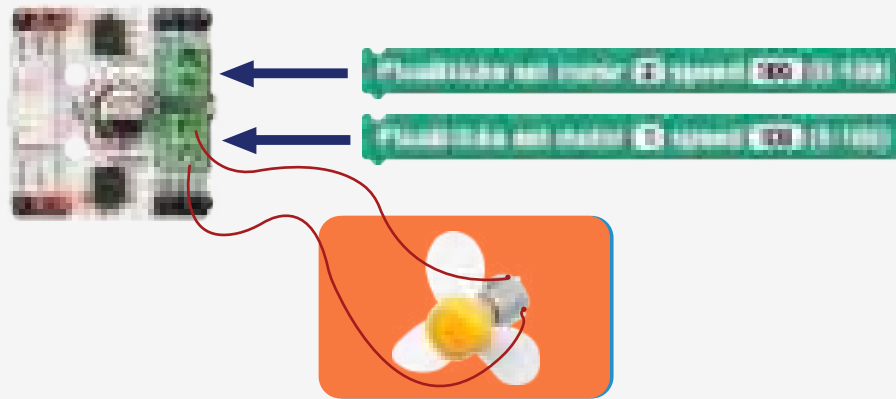



We will use the DC motor and propeller set in the Classroom kit for this activity.

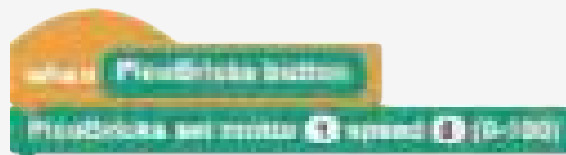
- 1.** Let's attach the propeller to the motor driver.
  - a.** As we will use a DC motor in the motor driver module, insert the black jumper so that the DC motor part is closed as shown in the image.
  - b.** Then, attach two cables on the motor of the propeller to the area reserved for the DC motor-1 with the help of a small screwdriver. (If you don't want to use a screwdriver, you can tighten them into these areas by using jumper cable ends.)



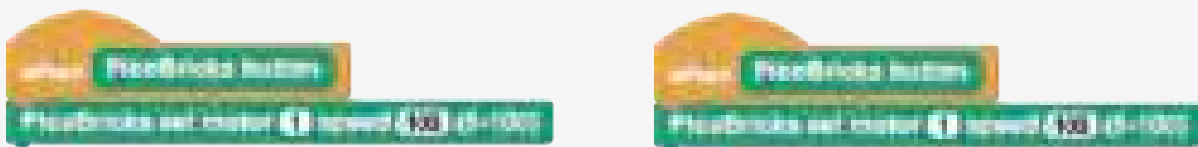
2. Now we can start dragging the code blocks. Let's drag the "when started" block and  the block that allows us to control the motor driver to our project page. There are 2 DC motor areas on the motor driver module. We should specify that we are attaching DC motor in the 1st or 2nd field in the code block as described in the picture below. Since we have attached the propeller in the 1st field, let's print the value 1 in the relevant field in the code block.



3. Let's drag the following code blocks to our project page to stop the propeller when we press the button. To stop the DC motor, we need to enter the "speed" value in the  block as "0".



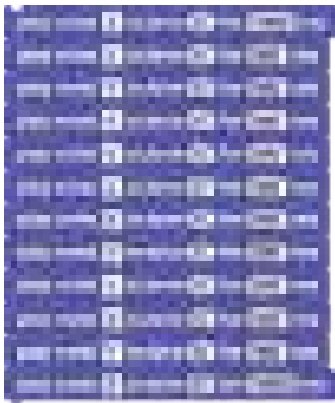
4. Our code is ready, now we can run our propeller by pressing the start button. To stop it, you must use the button on the PicoBricks.



## Project: Colorful Notes

Now, let's make a fun project by using the information we learned. In our project, we will use an OLED screen, button, buzzer, and RGB LED.

When the button is pressed, the notes given below will be played with the buzzer, while which note is working at the same time will be written on the OLED screen and the color of RGB LED will change simultaneously with the notes.



**Warning Box:**  
If you cannot find these blocks in MicroBlocks, you need to install the Tone library.

By dragging the above code blocks from the tone library to the project page, we can play the melody we want to play in our project with the buzzer. We can create our project by adding the RGB code in the color we want and the code block that allows us to write to the OLED between these tone blocks.

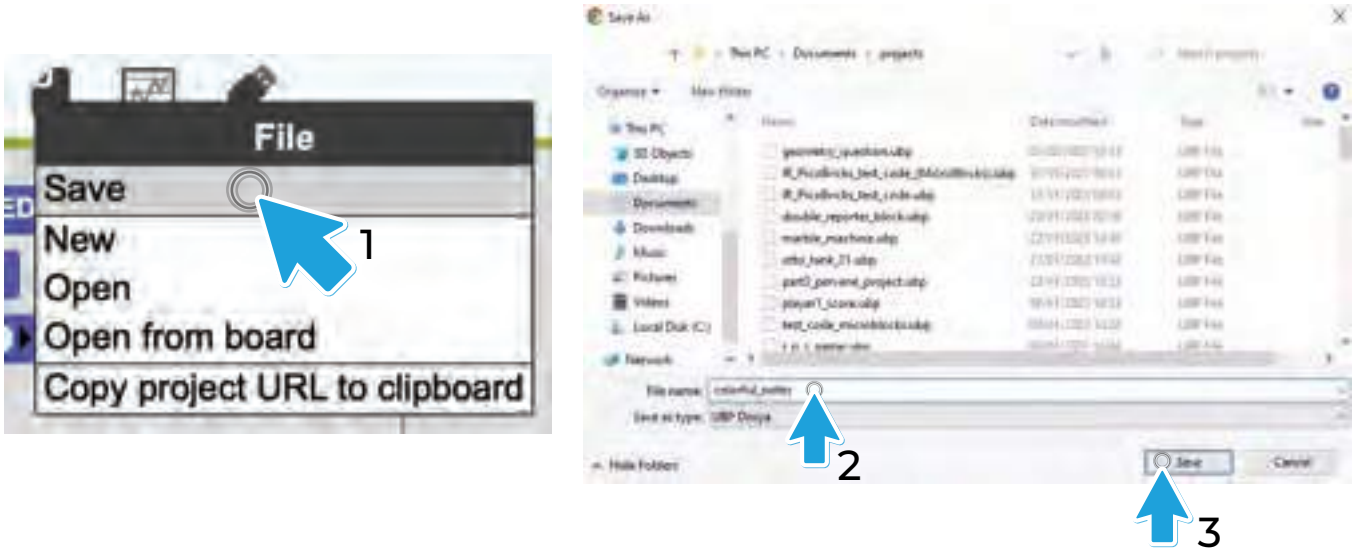
```
when PicoBricks button
  initialize I2c OLED_055a address(hex) 3C reset pin# 0 flip
  play note A octave 0 for 1000 ms
  write A at x 0 y 0 inverse
  PicoBricks set RGB LED color
  play note E octave 0 for 500 ms
  write E at x 0 y 0 inverse
  PicoBricks set RGB LED color
  play note E octave 0 for 500 ms
  PicoBricks set RGB LED color
  play note E octave 0 for 500 ms
  PicoBricks set RGB LED color
  play note E octave 0 for 500 ms
  PicoBricks set RGB LED color
  play note E octave 0 for 500 ms
  PicoBricks set RGB LED color
  play note E octave 0 for 500 ms
  PicoBricks set RGB LED color
  play note E octave 0 for 500 ms
  PicoBricks set RGB LED color
  play note D octave 0 for 500 ms
  write D at x 0 y 0 inverse
  PicoBricks set RGB LED color
  play note F octave 0 for 500 ms
  write F at x 0 y 0 inverse
  PicoBricks set RGB LED color
  play note E octave 0 for 1000 ms
  write E at x 0 y 0 inverse
  PicoBricks set RGB LED color
  play note F octave 0 for 500 ms
  write F at x 0 y 0 inverse
  PicoBricks set RGB LED color
```

After pressing the button, we need to drag these two code blocks to our project page in order to run the screen and other code blocks.

These are the code blocks that allow the tones required for the melody to be played from the buzzer, the tone value to be written on the screen and the color output from the RGB LED module. We can complete our project by copying these blocks and changing the values in them.

## Let's Save Our Project To The Computer

Let's save our project to the computer by following the steps below.



## K-12 International Computer Science Standards

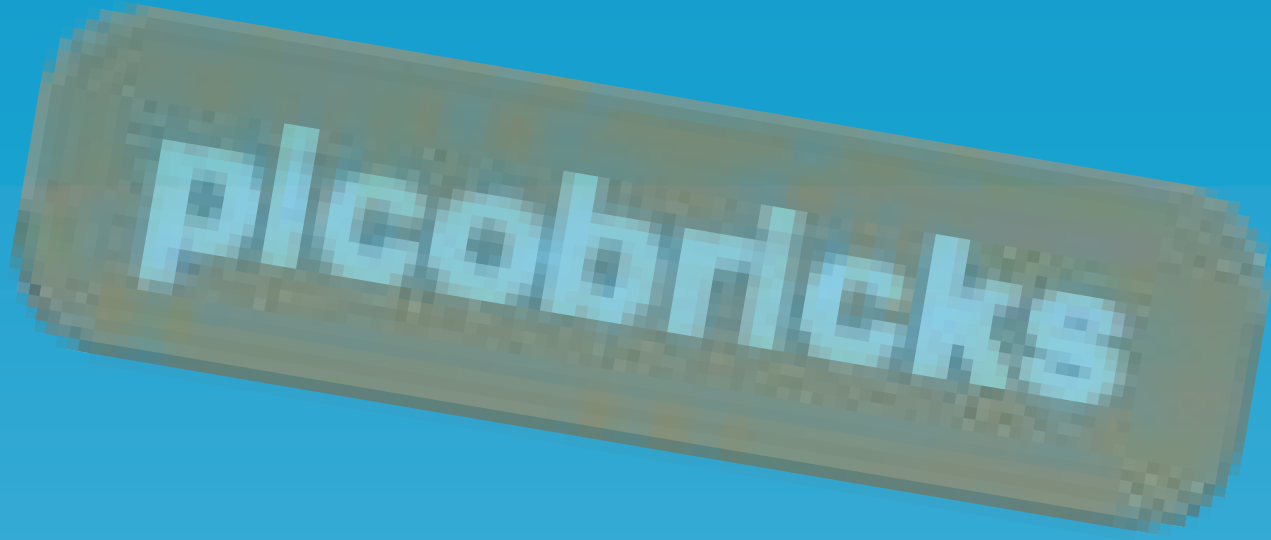
- 1B-AP-09-3-5 Create programs that use variables to store and modify data.
- 1B-AP-16-3-5 Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development.

set picobricks ▼ to 0



picobricks

# PART 3



picobricks

set picobricks ▼ to 0

## What We'll Learn in This Part

- In this part, the student learns to;
- Create variables on the MicroBlocks platform,
  - The definition of the variable,
  - The definition of the loop and its working logic,
  - Using the repeat loop with the MicroBlocks platform.

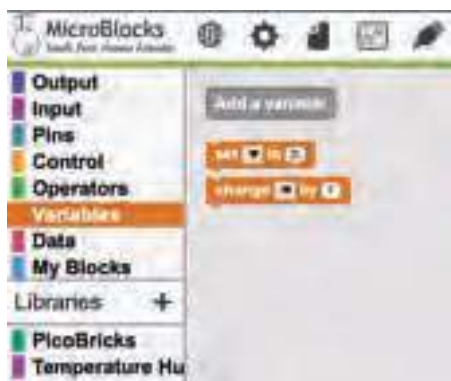
## Let's Define a Variable and Enter The Loop

### (x) What is a Variable?

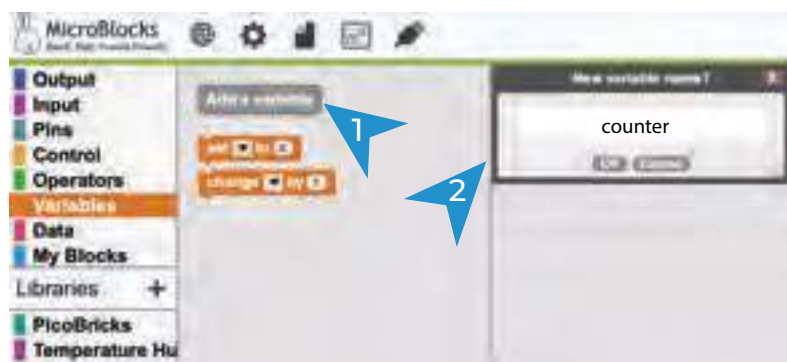
Variables are structures that store information that can be changed in the computer program. We can think of variables as containers that hold information.

Let's use the "wait" block to wait half a second between blocks

1. Click "Variables" (tab).



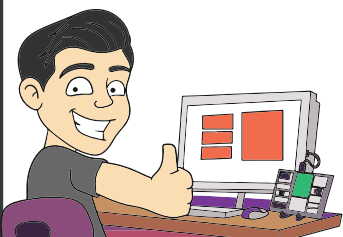
2. Let's specify the name of our variable by clicking on the "Add a variable" option. Here, we have defined the variable name as "counter" to be an example.





3. After specifying the variable name and pressing the "ok" button, the variable block we created comes between the "Variables" blocks. Now we can use the "counter" variable we created in our project.






## Now It's Your Turn!

Now let's create a variable named "PicoBricks" by following the steps above.

### Let's Set The Initial Value of The PicoBricks Variable

- Let's set the initial value of the PicoBricks variable to "0". When we start our project, we will use the  block of the "Variables" blocks to set the initial value of the PicoBricks variable. Let's change the value inside this block to PicoBricks and define the initial value as "0".






- After combining this code block we created with the "when started" block, we will set the initial value of the PicoBricks variable to "0".



## Let's Increase PicoBricks Variable by Pressing The Button

Each time the button is pressed, let's increase the value of the PicoBricks variable and print it on the screen. For this, let's drag the code blocks that start the screen every time the button is pressed and run the code blocks under it to our project page.



1. To increment the "picobricks" variable one by one each time the loop repeats Let's drag the block . To print the value of the "picobricks" variable on the OLED screen, let's start the OLED screen and drag the block  from the "Variables" blocks into the block. This code block we have obtained will allow us to print the PicoBricks variable to the screen. After this step, let's drag the block  to our project to reduce the reaction time. Now that the code blocks are formed, we can run our project. Each time we press the button, the PicoBricks variable will be written to the screen first, and then "1" will increase.



## What is Loop?

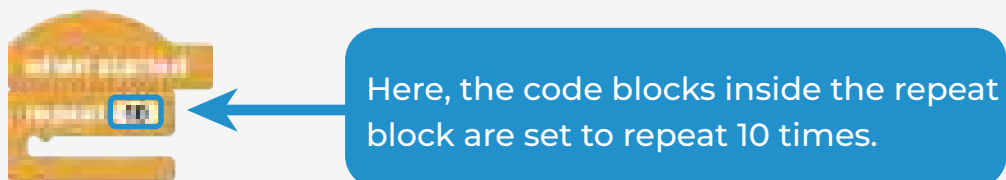
The process that repeats continuously until a certain condition is met is called a loop. For example, brewing tea, you put water on the stove and wait for it to boil, you need to check whether the water is boiling at certain intervals. This is a cycle.

In MicroBlocks, we use the  block to repeat the code blocks we drag into it as many times as we specify.

Let's turn on the red LED as much as the value we set using the repeat loop.

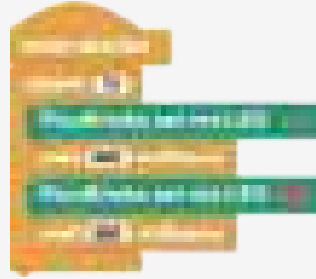
When we run the project from the project page, let's move the "when started" block to our project page so that the code blocks work.

- Let's drag the repeat block to our project page so that the loop starts right after the project is started. The number written on the repeat block determines how many times the code blocks inside the repeat block will repeat.

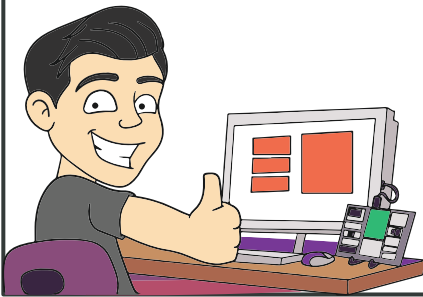




Now let's determine the code blocks that will repeat 10 times in the repeat block. After turning on the red LED on PicoBricks, let's wait for half a second and turn it off, then to better understand that the loop is repeating, let's put another half-second wait block at the end and drag these blocks inside the repeat block and let's run our project.



## Now It's Your Turn!



After determining the initial value of the variable named "PicoBricks" that we created, let's use the repeat loop to flash the red LED 5 times at half-second intervals.

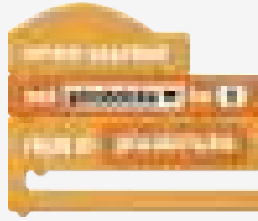
### Project Algorithm

1. Start.
2. Set the initial value of the PicoBricks variable.
3. Follow the steps below as much as the value of the PicoBricks variable.
  - a) Turn on the red LED
  - b) Wait for half-second
  - c) Turn off the red LED
  - d) Repeat half-second
4. Go back to the first step

- Drag the "when started" block from the control blocks to your project page. Then write the initial value to the PicoBricks variable we created before.



- Drag the repeat block to the project page and drag the PicoBricks variable inside the repeat value.



- Drag the required code blocks from the PicoBricks library and control blocks to flash the red LED at half-second intervals.



## Counter Activity

The counter activity aims to print numbers from 1 to 10 sequentially to the screen using variables and loops.

## Activity Image



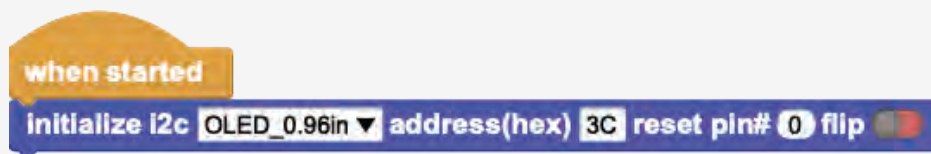
## MicroBlocks Code of The Activity

### Do not forget to install the OLED library!

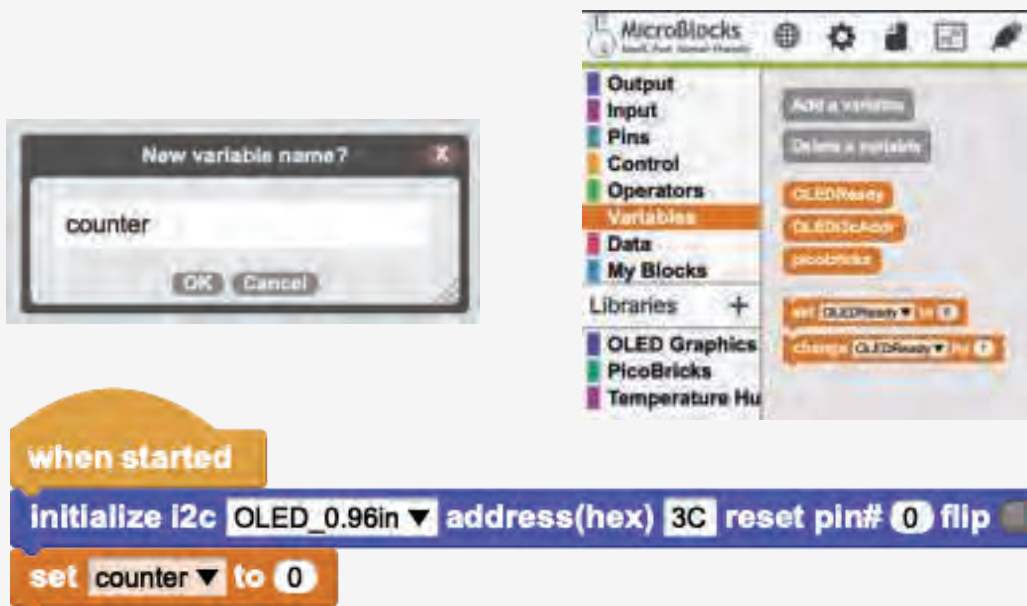
1. First, let's drag the "when started" block from the Controls blocks to our project page. In this way, after starting the program, other code blocks that we combined with this block will work.



2. Since we will use an OLED display in the event, let's drag the OLED display start block to our project.



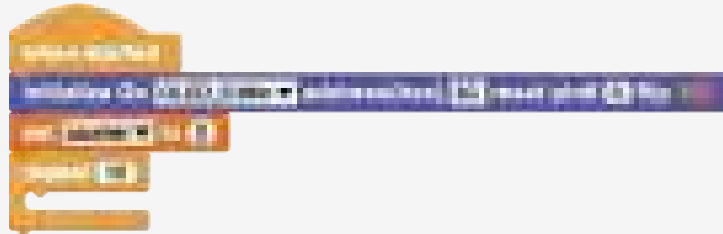
3. Let's create a counter variable and drag it to our project and set its initial value to 0.



### An Advice for The Teacher

What we have done up to this step consists of the information we have learned. After the necessary steps are explained to the students, the students are expected to carry out them without assistance. They can get help when it is necessary.

- Let's drag the repeat loop block from the control blocks to our project and set the number of repeats as 10. In this way, all the blocks we drag into the repeat loop repeat 10 times.



- We will use **Change Counter by 1** the block to increase the counter variable, which we have set as 0, one by one. After selecting the counter variable in the change block, set the increment amount to "1".



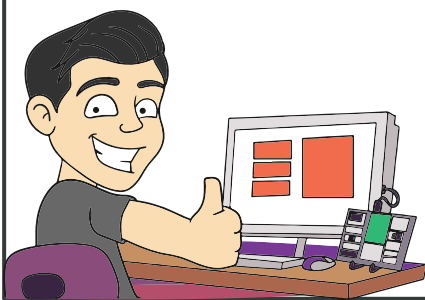
- Let's drag **Print Counter** the block from the OLED Graphic library to print the counter variable to the OLED display. To print the "counter" variable on the display, drag the "counter" variable block from the "Variables" blocks into this block.



5. Let's drag the wait 500 milisecs block to our project so that we can more easily observe the increase of the counter variable on the OLED display. In this way, the counter variable will change at half-second intervals.



## Now It's Your Turn!



Let's increase the "counter" variable by 2 and print it on the display. The last number on the screen must be "10".

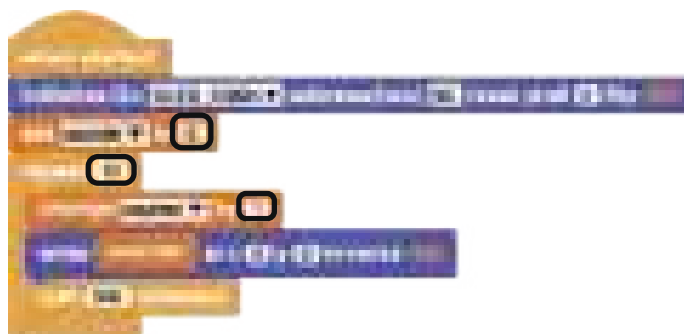
## HINT



We can realize this project by changing two different values in the code blocks of the activity we did before. Remember that the last number on the display should be 10!

## 2. Counter Activity

Let's do the reverse of this activity now. Let's drag the MicroBlocks code blocks, which print on the OLED display by reducing the numbers between 10 and 0 by 1, to our project page. **(Except 1 and 10.)** We do not need to rewrite the code to make this activity. After changing a few points in the first activity, the necessary code blocks for the activity will be ready. Let's examine these code blocks now. In the picture below, the points that need to be changed on the previous code blocks are marked.

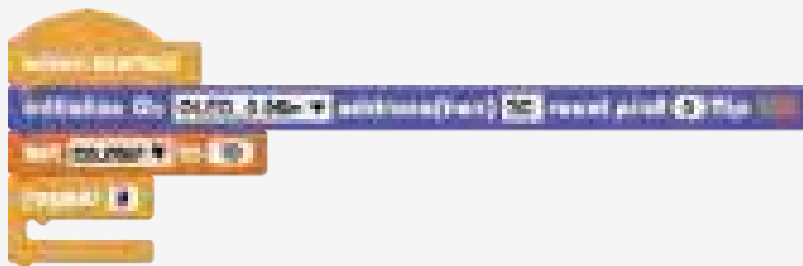


Since the first two code blocks are the necessary code blocks for the project and the OLED screen to start, we drag them to our project page in the same way.

After defining the variable "counter", let's set the initial variable as 10 since we will go backward from 10.



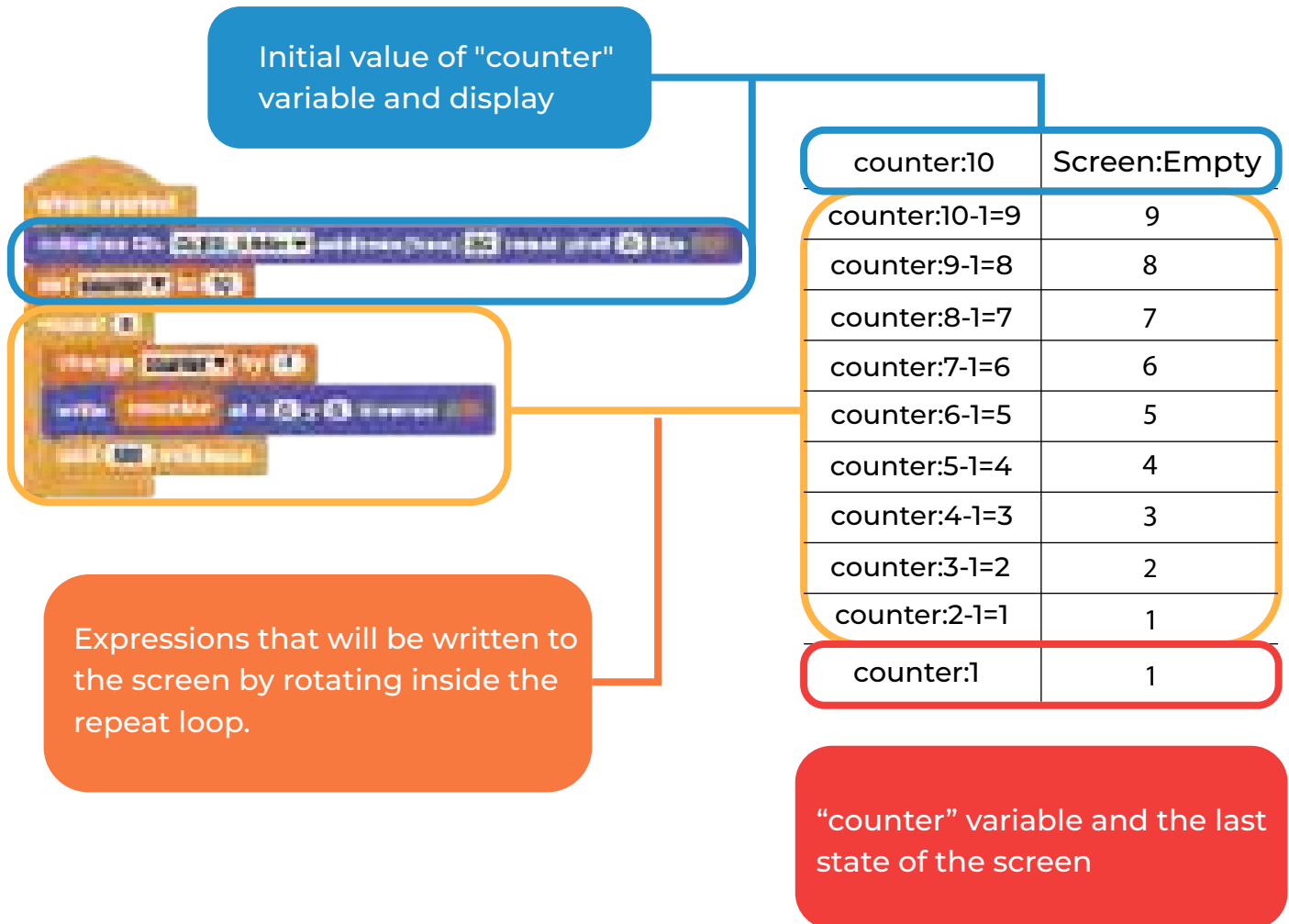
Let's define the repeat loop as 9 since we don't include 10 and 0.



For the "counter" variable to decrease by 1 each time the repeat loop repeats. Let's set the increasing amount as "-1" in the block.



Now let's examine how this code block we wrote works through the diagram below.



In the propeller activity you did in the second part, when you pressed the button, the propeller stopped and worked when you did not press the button. Now, let's make the propeller work when you press the button and stop when you do not.

**Solve the questions on the first worksheet!**  
**Solve the questions on the second worksheet!**



## K-12 International Computer Science Standards

- 2-AP-16 -6-8 - Incorporate existing code, media, and libraries into original programs, and give attribution.
- 2-AP-11 6-8 Create clearly named variables that represent different data types and perform operations on their values.
- 3A-AP-13- 9-10- Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
- 3A-AP-21- 9-10 - Evaluate and refine computational artifacts to make them more usable and accessible.



pen down



# PART 4

move 10



turn 90 degrees



pen up



## What We'll Learn in This Part

In this part, students learn;

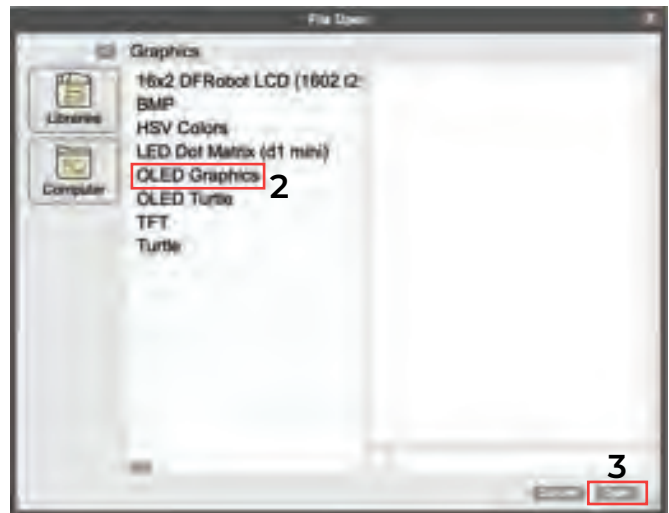
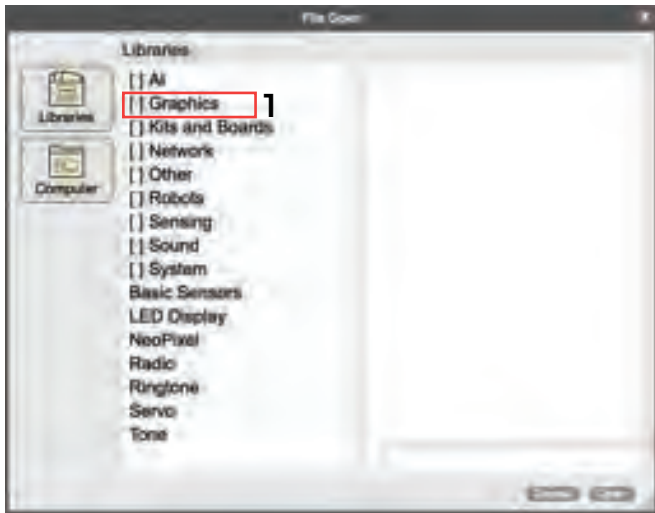
- To use loops in block-based programming tools,
- Create geometric shapes using loops,
- Develop projects with turtle library on MicroBlocks platform.

## Loops with Turtle

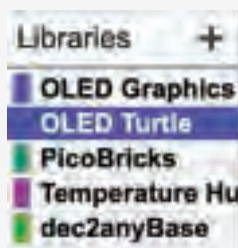
Turtle projects are a frequently preferred method in programming education. We can create turtle projects using the MicroBlocks turtle library on the OLED display, one of the PicoBricks modules

### Let's Install Turtle Library into MicroBlocks Editor





Let's install Turtle library to our project by following the steps given below.



We need to install OLED Graphics library too because we use Turtle library with OLED display.



Since we will be observing Turtle projects on the OLED display, we should definitely use the `initialize i2c OLED_0.96in address(hex) 3C reset pin# 0 flip` block that starts the OLED screen at the beginning of our project. Now let's learn the tasks of the code blocks that we will use for this section in the Turtle library.

	Block allows us to specify the Turtle size.
	Block allows us to draw a line on the path of the turtle cursor.
	Block determines how many steps the Turtle cursor will move forward in one move.
	Block determines how many degrees the Turtle cursor will rotate rotate at clockwise direction.

## Let's interpret the turtle project below

```

When started
initialize i2c OLED_0.96in address(hex) 3C reset pin# 0 flip
pen down
move 10
turn 90 degrees
move 10
turn 90 degrees
  
```

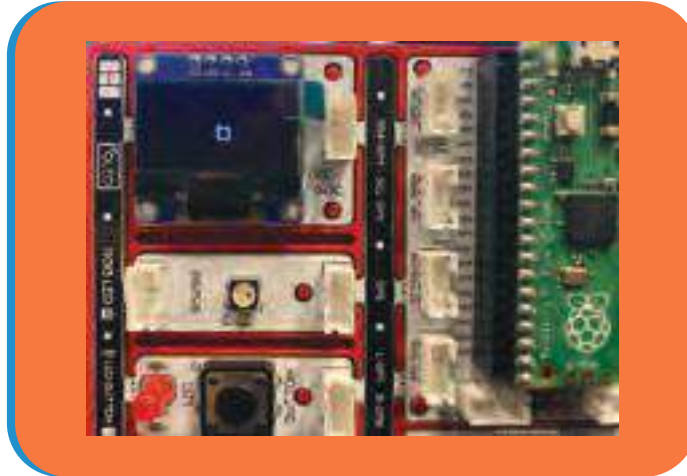
- Initialize OLED display.
- Draw a line on the path of the Turtle cursor.
- Move the Turtle cursor 10 steps forward.
- Rotate the Turtle cursor 90 degrees.
- Move the Turtle cursor 10 steps forward.
- Rotate the Turtle cursor 90 degrees.



## Some Shapes with Turtle

We can create various geometric shapes by changing the degree value of the `turn 90 degrees` block in the Turtle library on the OLED display. Let's take a look at how we can create these geometric shapes;

### SQUARE



Let's assume that the Turtle cursor moves in only one direction (We can choose this direction as right). A cursor that constantly rotates 90 degrees to the right will have moved in a square shape when it repeats this movement 4 times. If we draw the paths it follows in this process with a pencil, we will observe the square shape.



In order to follow the movements of the Turtle cursor, we need to drag the above code blocks to our project page. Let's explain these code blocks in order;

- It allows us to run the code blocks that come under after starting our project.
- Since we will use the Turtle cursor with the OLED display, the OLED display is the code block that allows us to start.
- The code block that determines the size of the Turtle cursor.
- It is a block of code that allows us to follow the path of the Turtle cursor.

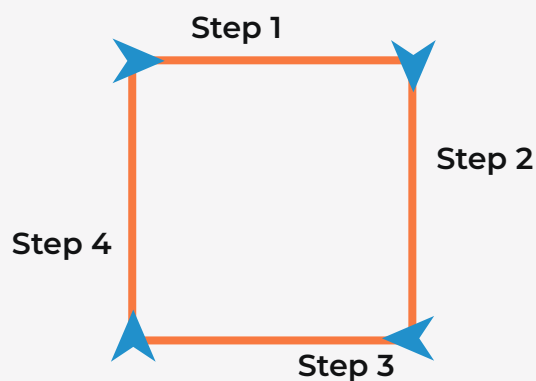
Now let's rotate the Turtle cursor 90 degrees so that it follows a square path. You can increase or decrease the edge length by changing the number of steps.



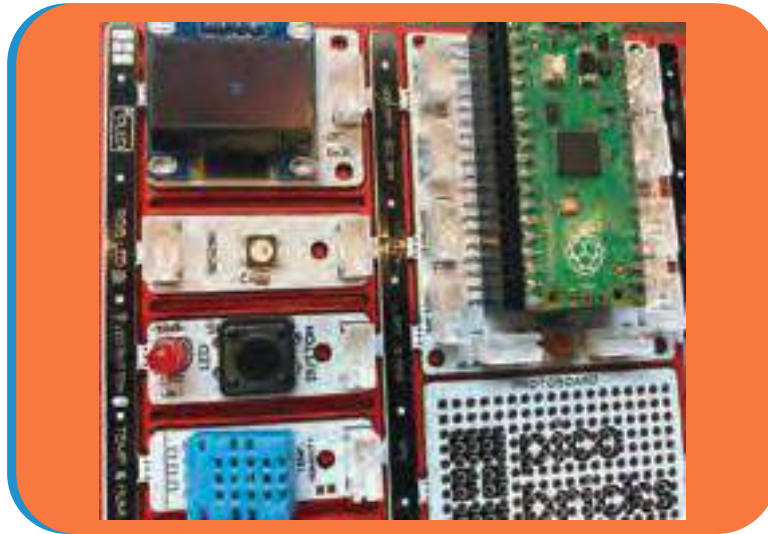
If we combine the above code block 4 times in a row, we will observe a square with a side length of 10 steps on the display.



The diagram below shows the steps that turtle cursor follows.



# TRIANGLE

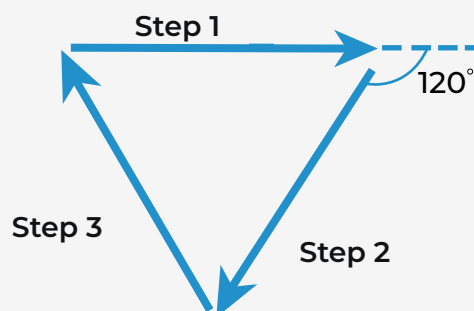


If the turtle cursor moves at an angle of 120 degrees 3 times, it will move in a triangle. Let's create a triangle.

```

when green flag clicked
  initialize (0) [0.000000] address (hex) [ ] read (port) [ ] flip
  initialize turtle [ ]
  pen down
  move [10]
  turn [120] degrees
  move [10]
  turn [120] degrees
  move [10]
  turn [120] degrees
  
```

Let's examine movements of the turtle cursor and the code blocks above.



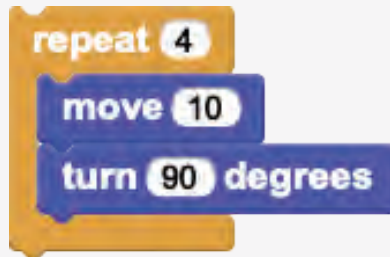


We created the square and triangle shapes on the OLED display by repeating the same blocks as the number of steps in the shapes we created with Turtle above. We can also create the same shapes using the “repeat” block. Let's create the square shape on the OLED display using the repeat block;

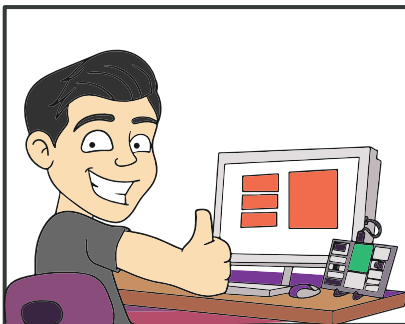
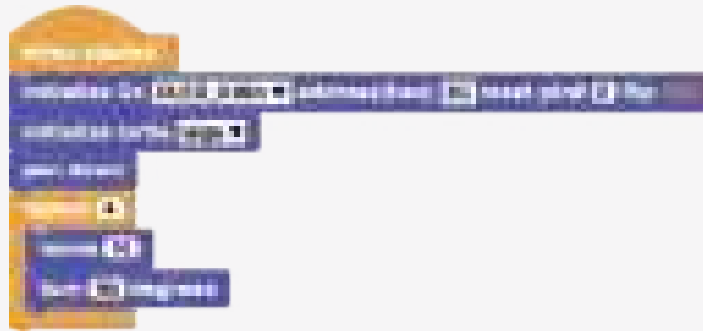
In the code blocks that we created the square shape before, we created the shape by bringing the following two code blocks four times in a row.



Now let's drag this code block into the repeat block and make it repeat block and make it repeat 4 times.



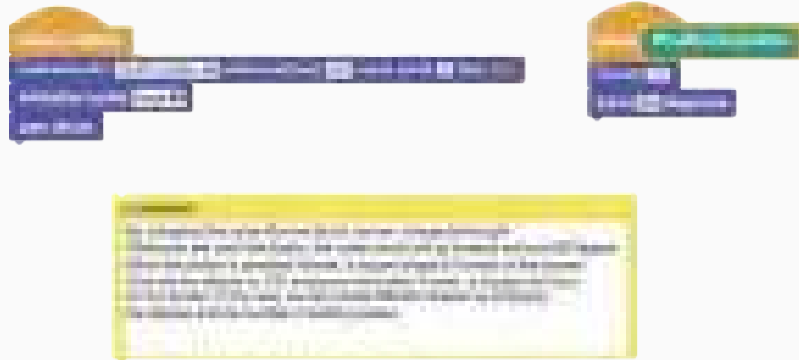
We set the value of the repeat block is 4 so that the code blocks in it repeat 4 times.



## Now It's Your Turn!

Create a triangle shape with turtle cursor by using “repeat” block.

## Some Advices For The Teacher



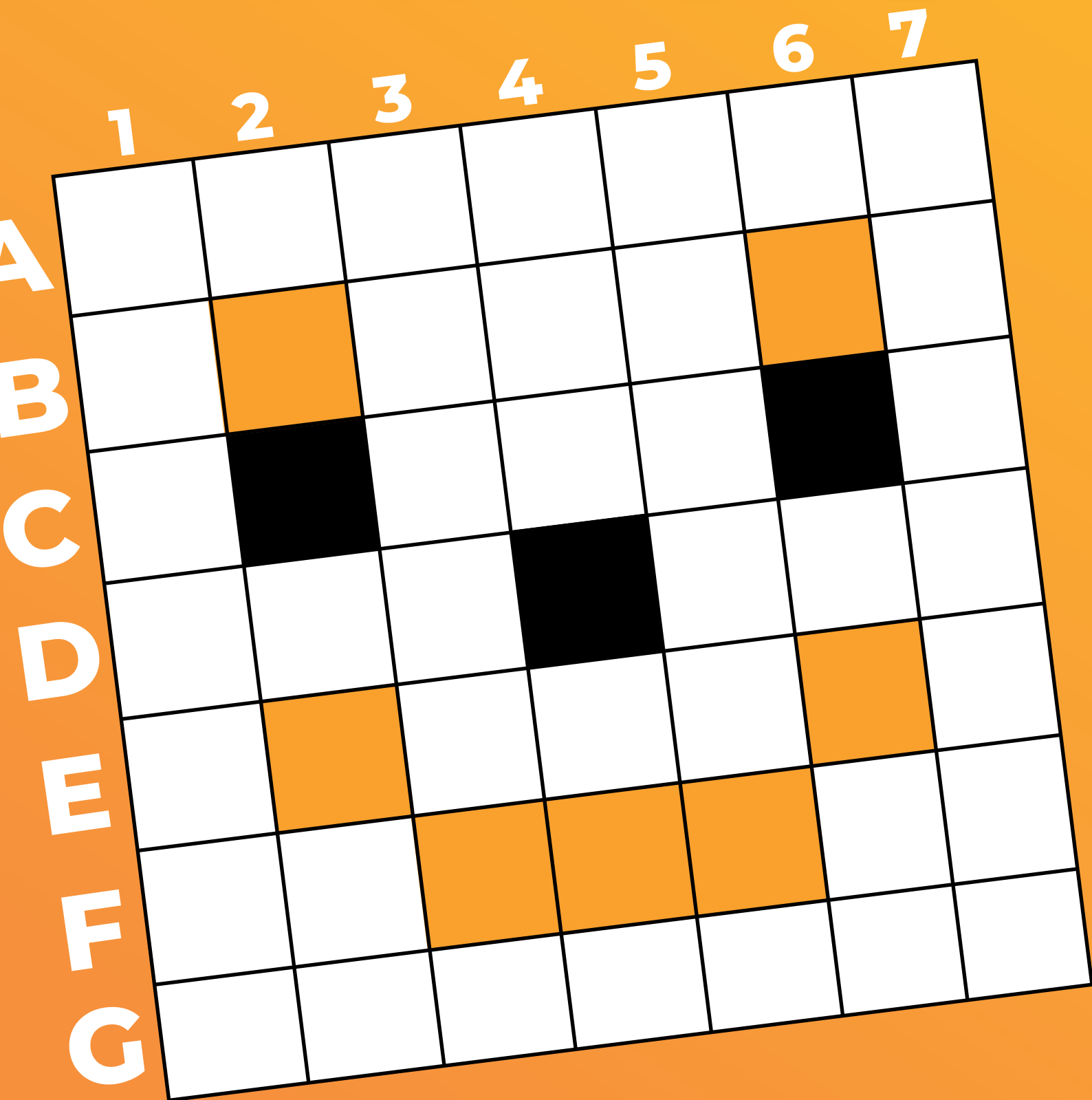
You can use the code blocks above to explain the repetitions of the moves and the loop logic before starting this activity. For example, after making the value of degree is 90, by pressing the button 4 times, a square shape can be created on the screen with the turtle cursor. By making the degree 120, the students were asked, "How many times do I press the button, a triangle shape will appear on the display?", "What shape is formed when I press the button three times?" etc. The working logic of the loops can be explained by asking questions.

**Solve the questions on the worksheet!**

## K-12 International Computer Science Standards

- 1B-AP-09 3-5 Create programs that use variables to store and modify data.
- 1B-AP-10 3-5 Create programs that include sequences, events, loops, and conditionals.
- 3A-AP-13 9-10 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.

# PART 5



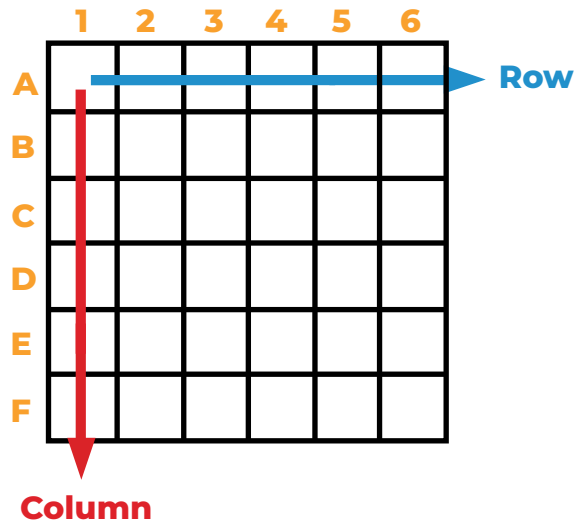
## What We'll Learn in This Part

In this part, students learn;

- Determining coordinates using rows and columns,
- Positioning logic of the PicoBricks OLED display module.

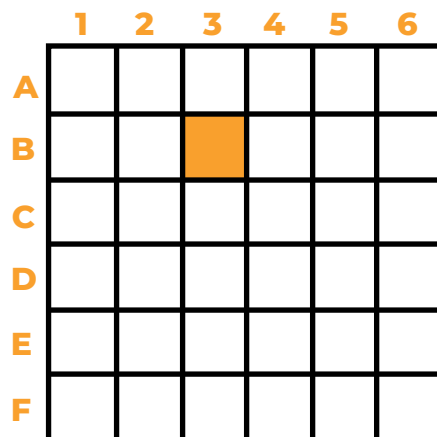
## Unplugged: Let's Play Admiral Sunk

Let's learn the coordinate system by playing the admiral sunk on the 6x6 square playground.



Before starting a real admiral sunk game, let's get to know the 6x6 playing field. Each square on the playing field has its own coordinate point. For example;

The coordinate of the point marked in orange on the 6x6 playing field below is **B3**.



Areas indicated by letters represent rows and areas indicated by numbers represent columns. The point marked on the playing field above is in row B and column 3. Therefore, the coordinate point is **B3**.

- Solve the questions on the worksheet below.

	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>
<b>A</b>						
<b>B</b>						
<b>C</b>						
<b>D</b>						
<b>E</b>						
<b>F</b>						

According to the adjacent table, which of the following coordinate points is not colored orange. Indicate by coloring the circle given next to it.

- |                       |           |                       |           |
|-----------------------|-----------|-----------------------|-----------|
| <input type="radio"/> | <b>A5</b> | <input type="radio"/> | <b>D4</b> |
| <input type="radio"/> | <b>B3</b> | <input type="radio"/> | <b>A2</b> |
| <input type="radio"/> | <b>C3</b> | <input type="radio"/> | <b>D2</b> |
| <input type="radio"/> | <b>B4</b> | <input type="radio"/> | <b>C4</b> |

	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>
<b>A</b>						
<b>B</b>						
<b>C</b>						
<b>D</b>						
<b>E</b>						
<b>F</b>						

According to the adjacent table, fill in the blanks at the coordinate points given below.

- |                       |           |                       |           |
|-----------------------|-----------|-----------------------|-----------|
| <input type="radio"/> | <b>D_</b> | <input type="radio"/> | <b>E_</b> |
| <input type="radio"/> | <b>C_</b> | <input type="radio"/> | <b>D3</b> |
| <input type="radio"/> | <b>C5</b> | <input type="radio"/> | <b>E3</b> |
| <input type="radio"/> | <b>B_</b> | <input type="radio"/> | <b>F_</b> |

	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>
<b>A</b>						
<b>B</b>						
<b>C</b>						
<b>D</b>						
<b>E</b>						
<b>F</b>						





Indicate the coordinate points given below by coloring the correct places on the adjacent game table.

- |                       |           |                       |           |
|-----------------------|-----------|-----------------------|-----------|
| <input type="radio"/> | <b>A2</b> | <input type="radio"/> | <b>D2</b> |
| <input type="radio"/> | <b>B2</b> | <input type="radio"/> | <b>D5</b> |
| <input type="radio"/> | <b>E3</b> | <input type="radio"/> | <b>A5</b> |
| <input type="radio"/> | <b>E4</b> | <input type="radio"/> | <b>B5</b> |

- Now, let's play the admiral sunk on the 6x6 playground.

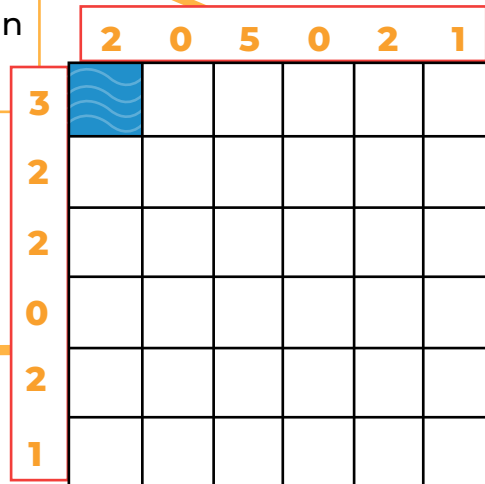
Now, that we know the playground, let's see how the admiral sunk game is played.

Let's learn the admiral sunk game through the example of the admiral sunk game given below

<ul style="list-style-type: none"> <li>• It represents the sea, when you see this blue box on the playground, it means there are no ships in those coordinates.</li> </ul>	
<ul style="list-style-type: none"> <li>• There are types of ships of 1, 2 and 3 units in the admiral sunk game we'll play:</li> </ul>	
<ul style="list-style-type: none"> <li>◦ Represents a one-unit ship.</li> </ul>	
<ul style="list-style-type: none"> <li>◦ Represents a two-unit ship.</li> </ul>	
<ul style="list-style-type: none"> <li>◦ Represents a three-unit ship.</li> </ul>	

The numbers at the beginning of the columns indicate how many square units are filled in the column.

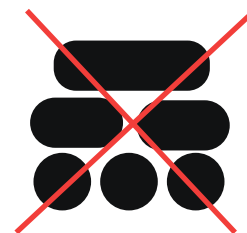
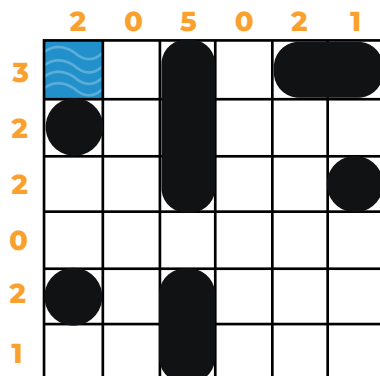
The numbers at the beginning of the lines indicate how many unit squares are filled in the line.



It represents the ships that must be placed on the playing field. This means there are: one 3 unit, two 2 unit, three 1 unit in this game.

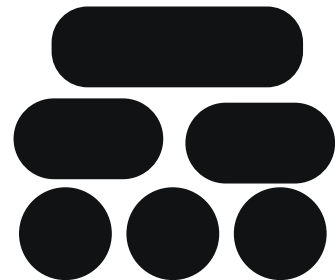
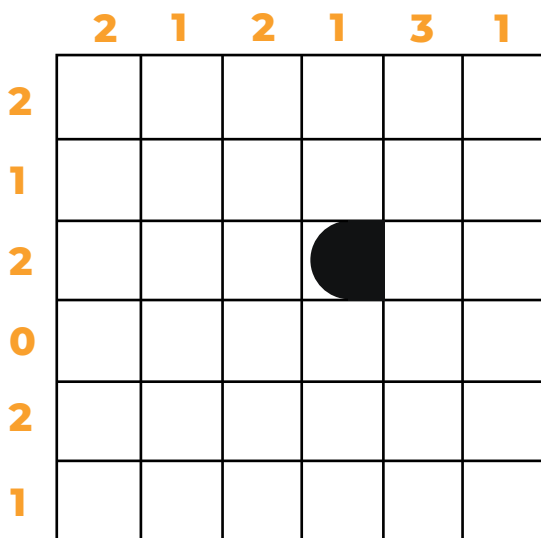
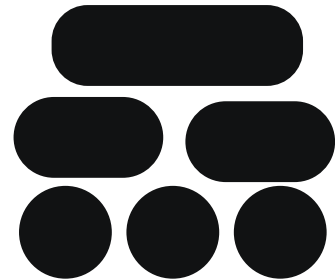
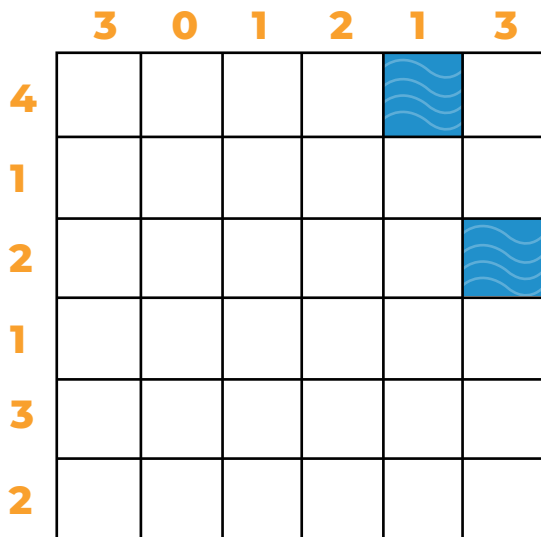
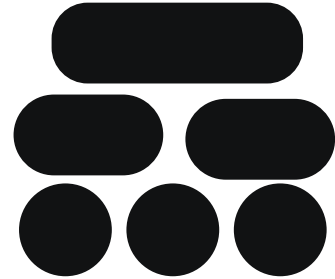
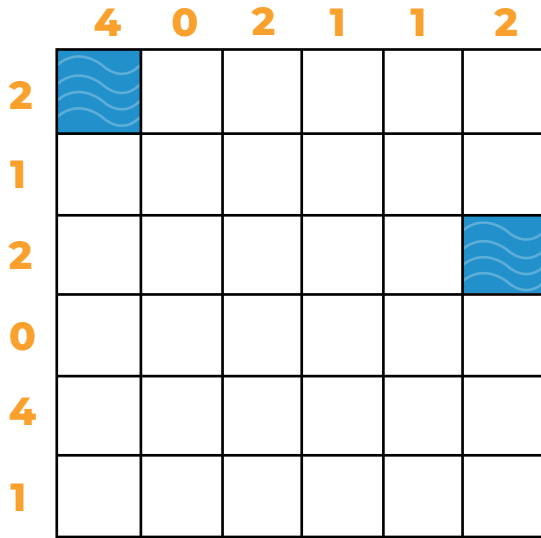
- Let's play the admiral sunk game that is given above

We left this area blank because it's the sea.



We have placed all the ships on the playground according to clues at the beginning of the rows and columns.

# Admiral Sunk Worksheet

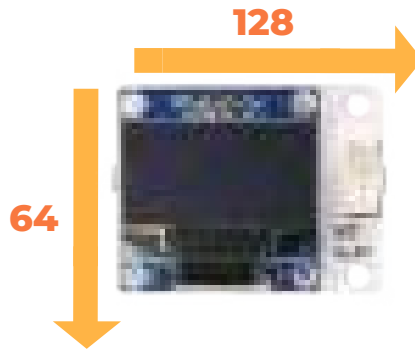


(In some games, some of the ships can be given as hints.)




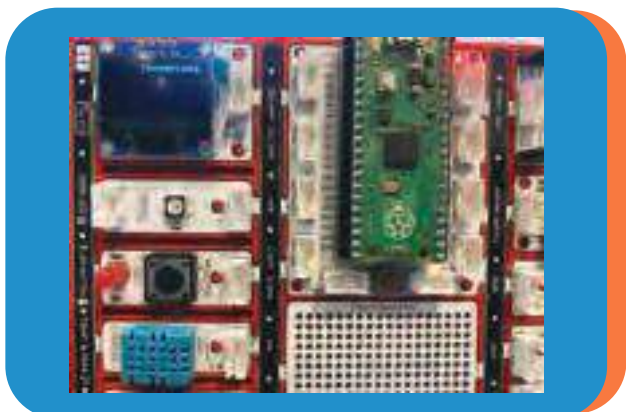
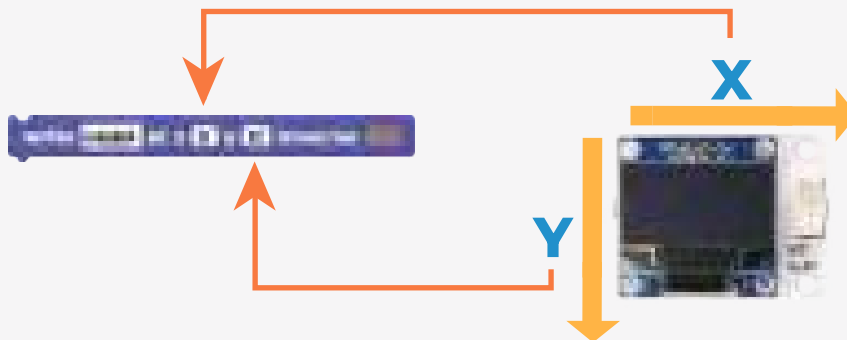
## Coordinate Points on the OLED Display

The size of the OLED display module on PicoBricks is 128x64 as shown in the diagram below. In order to determine the position of the visual or textual output that we want to receive on the screen, we need to value according to 128x64 dimensions. 64 represents the height of the OLED screen and 128 the width.

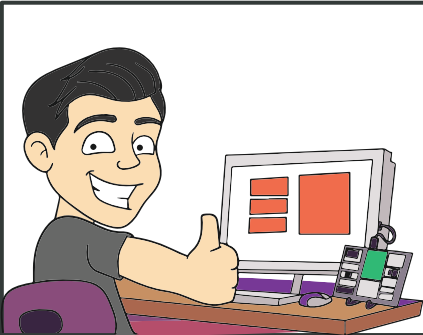


Let's adjust the position of the text written on the OLED screen module.

According to the information we have learned before, we use  the block to write a textual expression on the OLED screen. We can change the position of the expression we want to print on the screen by changing the x and y values on this block. The x value represents the width of the OLED screen and the y value represents the height.



Let's print "PicoBricks" on X=45, Y=0 coordinates.



## Now It's Your Turn!

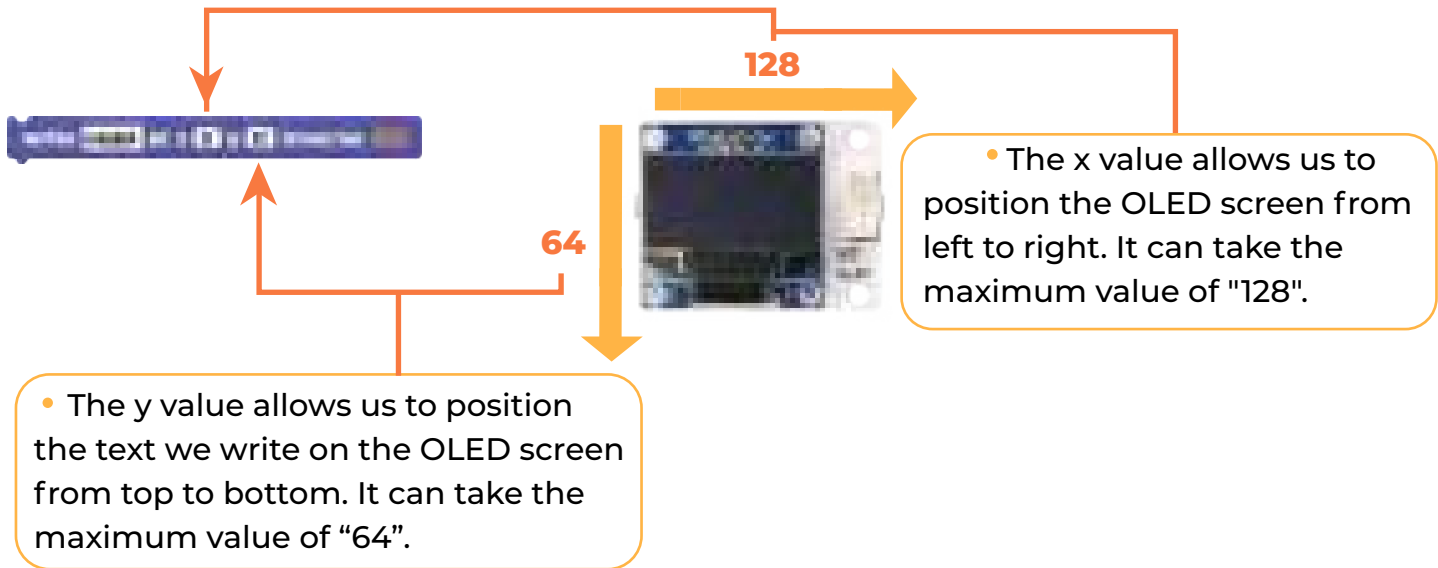
Now, print your name on the coordinates X=0, Y=10.

### PicoBricks Text Scrolling From Top To Bottom



Now, let's slide the text "PicoBricks" from the top of the OLED screen to the bottom. After dragging the blocks that enable the project and the OLED screen to start on our project page, let's define a variable named "scroll" and set the initial value to "0".



Before writing the next code blocks, let's repeat the OLED screen features through the visual diagram below.

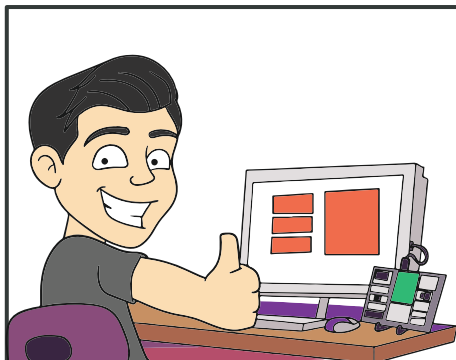


Let's write the code blocks, that allow us to scroll "PicoBricks" text down 10 units on the OLED screen.

Let's drag the "Repeat loop" block to our project page and set its value as "7". Since the maximum value that the screen can take from top to bottom is 64 and the maximum value that the "y" value in the  block can take in our loop is 60, we repeated the loop seven times. Let's drag the "scroll" variable to the point where the "y" value is located and change the expression that will be written on the screen to "PicoBricks". After waiting for 50 ms,  let's add 10 to the variable x and clear the screen every time.



## The Project is Ready!



## Now It's Your Turn!

Let's create the code blocks that ensure to slide "x" from left to right.

## Activity

Discuss in class how the following code blocks will produce an output on the OLED screen. Then, drag the code blocks to your project page and observe the change on the OLED screen on PicoBricks.



## The Project

Let's do the thermometer activity we did before, but this time let's change the expressions on the OLED display module a little. In the thermometer activity we did before, we were printing the value we received from the temperature and humidity sensor on the OLED screen at the coordinates X=0, Y=0, this time we print the expressions "Temperature:", Temperature value, and "degree" on the OLED screen and get an output like the PicoBricks given in the picture below.

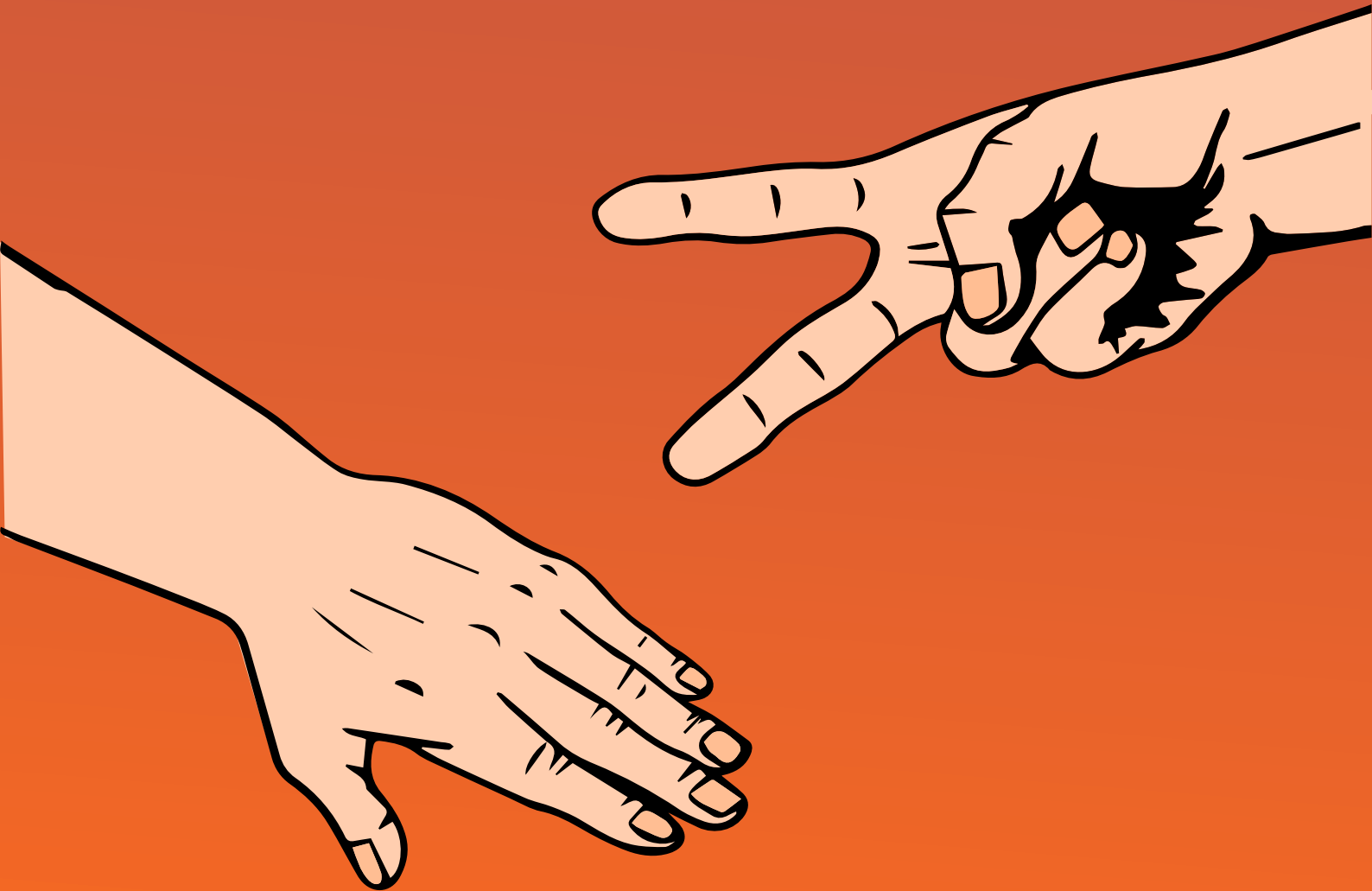


We will print the expression "Temperature" at X=15, Y=10, the temperature value taken from the temperature and humidity sensor at X=55, Y=30 and the expression "degree" at X=40, Y=50.



## K-12 International Computer Science Standards

- 1B-AP-12 3- 5 Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.



# PART 6




## What We'll Learn in This Part

- In this part, the student learns to;
- If-else structure from condition statements,
  - The structure of forever from the loop statements,
  - Creating a random value.

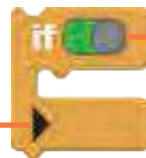
## Introduction to Conditional Structure

### IF-ELSE

If - else is a conditional statement. Conditional statements are statements that allow the desired action to be performed when a certain condition is met. We often use these expressions in our daily life. For example, we boil water to brew tea, and turning off the stove after the water boils is a conditional task. If the temperature of the water exceeds 100, we will have to turn off the stove. We also use conditional statements to perform conditional operations on programming platforms. In this section, we will develop projects using the if-else statement. In MicroBlocks,  the block is used for if-else. The purpose and usage of this code block are explained in the diagram below.

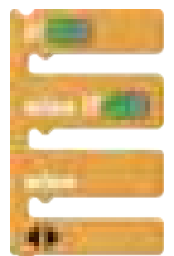
Clicking this cursor opens else block. Else block determines the actions to be taken when the statement inside the If block is not executed.

The desired condition expression is brought to this field.



Clicking this cursor returns the if-else block to its previous state.

The else if block, which we click on the cursor again, opens between the if and else blocks. The purpose of the else if block is to perform different condition conditions than the if block and the actions to be performed after the specified conditions are met. The more we on this cursor, the more else if blocks open.





## Unplugged: Activity

Rock-paper-scissors activity is a game played by hand. We get points according to our rock-paper-scissors movements and we try to defeat our opponent as a result of these points.

- When we open our hand completely, it is paper.
- When we make a punch, it is rock.
- The shape, we create by closing our fingers except for the index and middle fingers, represents scissors.



The winner in this game is determined in this way:

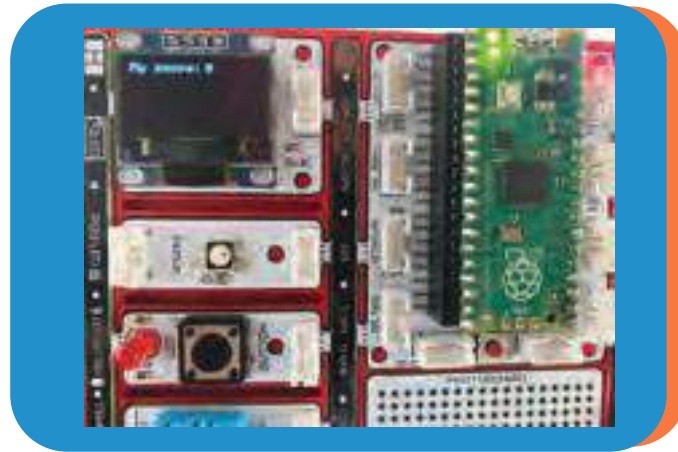
- Rock breaks scissors. In this case, the person who makes rock defeats the person who makes scissors.
- Scissors cut paper. In this case, the person who makes scissors defeats the person who makes paper.
- Paper wraps rock. In this case, the person who makes paper defeats the person who makes rock.

## Let's Play Rock-Paper-Scissors Five Times With Your Friend

- Let's play rock-paper-scissors with your friend/friends and write down your scores on paper.
- Now, create your own leaderboard with PicoBricks and play rock-paper-scissors with your friend again and note your score on PicoBricks.
- Now, everyone has their own score written on their PicoBricks. Follow the project steps below to save your score to PicoBricks.

## Score Activity

The purpose of this activity is to record the points obtained by the student in the game by using the PicoBricks. Each time the student presses the button, their score will increase by one and will be written on the OLED screen. Let's examine the project algorithm and drag the code blocks to our project page.



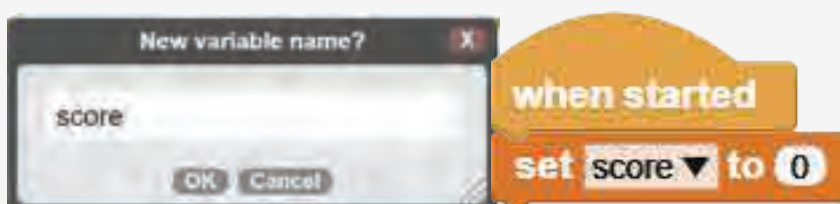
### Project Algorithm


1. Start.
2. Set the initial value of the score value to "0".
3. If the button is pressed, increase the score value by 1 and print it on the OLED screen.
4. If not, print the current state of the score value to the OLED screen
5. Repeat step 4 constantly..

## MicroBlocks Code of The Project

Now let's start dragging the code blocks.

Let's define a variable named "score" after the "when started" block and set the initial value of this variable to "0".



After this step, we will use a loop block to repeat all the operations we will do until the program ends. We learned about the repeat block from loop statements before. In this section, we will learn about block .  It ensures that all code blocks dragged into the Forever block continue until the program is terminated.





Now, let's determine the statements that will continue until the program is terminated in the forever block. Since we will use the OLED screen, let's drag the

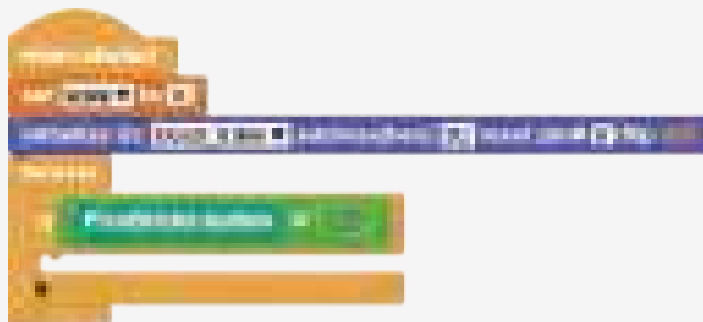
 block that starts the OLED screen into the forever block.



Now let's start dragging the condition statements into our project page. Let's drag the if block to our project page. The if block ensures that the code blocks inside are executed if the condition is met.



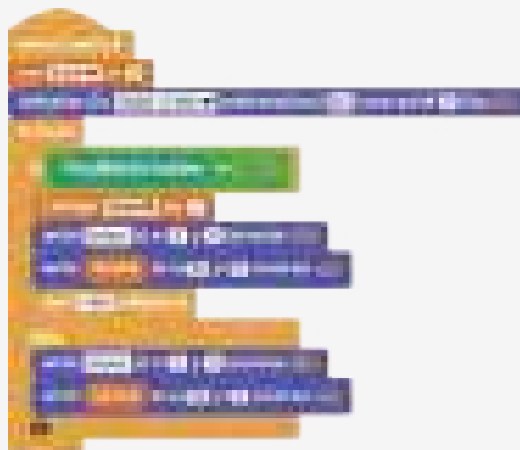
Let's determine the condition expression inside the if block. Let's create the code block that allows us to understand that the button was pressed to print the score variable once, when the button is pressed, and drag it into the if block. In the equals operator,  let's drag the  block from the PicoBricks blocks to the left side of the equation and the  block to the right side of the equation, after creating the  code block, let's drag it into the if block.



Let's determine the events that will happen if the condition is true, that is, when the button is pressed. When the button is pressed, we want the "score" variable to be increased by one and then written to the screen with the text "Myscore". Let's drag the code blocks that provide this into the if block. Let's print the expressions that will be written on the screen with X and Y coordinates on the coordinate points we want.

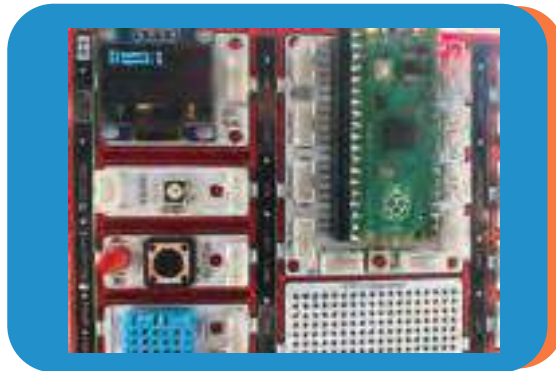


Let's open the else block by clicking the cursor under the if block. Let's drag the statements that will happen when the statement in the if block is not executed, that is, when the button is not pressed, into the else block. When the button is not pressed, we want the current score variable and "Myscore" text to be displayed on the screen.



We can run our project and continue playing rock-paper-scissors. Now we can use PicoBricks instead of notepad.

## Extra Project: Two-Player Score Activity



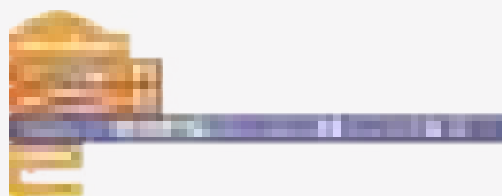
In the score activity we did before, each student was writing their own score on their Picobriks, now we can increase the score of both players by using a single PicoBricks. In the previous activity, we used the button module to increase the "score" variable. Since both players will use the same PicoBricks in this event, you will need to use two different modules to increase the score. In this activity, player 1 will use the LDR sensor module and player 2 will use the button module to increase their scores.

### Project Algorithm

1. Start.
2. Initialize the player 1 variable to "0".
3. Initialize the player 2 variable to "0".
4. If the ambient light amount is small, increase the player1 value by 1 from 50 and write it on the screen.
5. If the Button is pressed, increase the player 2 value by 1 and print it on the screen.
6. If the possibilities in the 4th and 5th items are not fulfilled, write the current value the player 1 and player 2 variables to the screen.
7. Repeat step 4, until the program is terminated.

### MicroBlocks Code of The Project

Let's start dragging the necessary code blocks for the project to our project page. Since two students will use a single PicoBricks in the project, we define the "player1" and "player 2" variables for each student, that is, the player, and enter the initial values of these variables as "0". In the same way, let's drag the forever block just below these blocks so that the other operations we do continue until the program is terminated. Since we will use the OLED screen, let's drag the OLED screen initialization block into the forever block.



Now let's define our condition statements and drag the necessary code blocks.

Since two players will increase their scores with two different conditions, if - else if and else blocks will be used in this project. The 1st player will cover the LDR sensor with his finger and increase the variable score variable, for this, let's write the code block expressing the sentence "if the value of the LDR sensor is less than 15" and drag it into the if value. Let's write.



Then, in the if block, increase the value of the "player1" variable by one and print the values of "player 1:" and player1 variable to the coordinates determined on the OLED screen.



Since the second player will increase the score variable when the button is pressed, we create a code block that expresses this situation and drag it into the else if block.



Then, let's increase the "player 2" variable by one and print the player 2 variable and the text "Player 2:" on the coordinates determined on the OLED screen. In the Else block, let's write the statement that will appear on the screen when these two conditions are not met. That is, in cases where the button is not pressed and the LDR sensor is not covered, the OLED screen Let's bring up the code blocks that print the current values of the player1 and player 2 variables.

Now we can run our project and note our scores with the same PicoBricks.

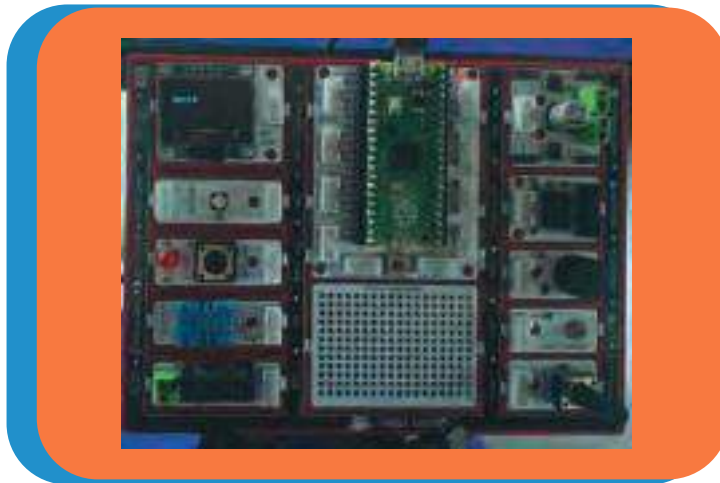


## Images of the Activity Rock-Paper-Scissors MicroBlocks Project

1. Press the button and wait 3 seconds.



2. One of the Rock-Paper-Scissors will randomly appear on the OLED screen.



3. One of the Rock-Paper-Scissors will randomly appear on the OLED screen.



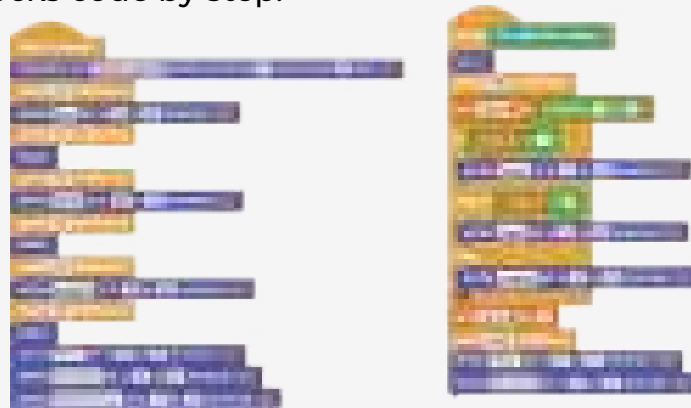


## Project Algorithm

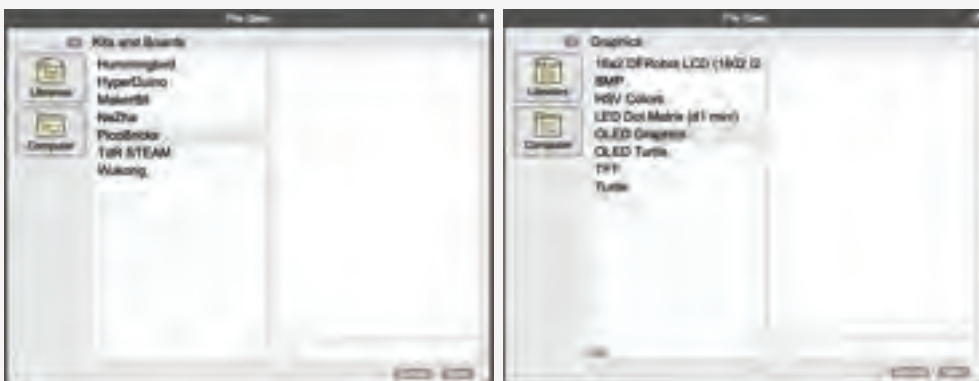
1. Start.
2. Print "Rock","paper","scissors" to the screen at 100ms intervals.
3. Clear the screen and print "press", "the button", "wait the 3 secs" on the screen.
4. When the button is pressed.
5. Clear screen and wait 3 seconds.
6. Define a variable named r\_p\_s and assign a random number between 1 and 3 to this variable.
7. If the number is 1, print "Rock" on the screen.
8. If the number is 2, write paper on the screen.
9. If the number is not 1 or 2, print "scissors" on the screen.
10. Then reset the r\_p\_s variable
11. Wait 3 seconds and return to item 1 if the button is pressed.

## MicroBlocks Code of The Project

The entire MicroBlocks code of the project is given in the image below. Now, let's examine the MicroBlocks code by step.



1. Let's install the PicoBricks library and OLED Graphics library into our project for the Rock-Paper-Scissors activity.





Now let's define a variable named `r_p_s` as we learned earlier. We will then use this variable to print one of the random Rock-Paper-Scissors to the OLED screen. To make the initial value of the `r_p_s` variable a random number between 1 and 3, let's drag the operator blocks into `random(1, 4)` of the `set_variable_value` block and set the value "10" to "3". Thanks to this `set_variable_value` block we have created, the `r_p_s` variable will get a random one between 1 and 3 every time a button is pressed

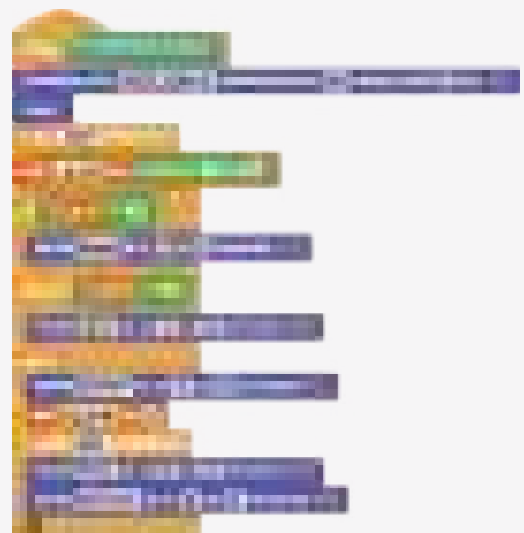


Now we can move on to conditional statements. Let's create if, else if and else blocks to create code blocks that print the value of "Rocks" if the `r_p_s` variable takes the value "1", "Paper" if it takes the value of 2, and "Scissors" if it does not take the value of 1 or 2, that is, if it takes the value of 3. Let's fill in conditional expressions such as:



To replay the game right after these blocks, let's set the value of the `r_p_s` variable back to "0". After waiting for three seconds, let's print the expressions "press" and "the button" on the screen again.

We can run the project and play rock-paper-scissors against PicoBricks.



### Some Advice For The Teacher

- The steps given above should be performed together with the students.
- While explaining the variable identification phase, students should be asked to define the variable without assistance by making use of the previously learned information. Students who have difficulties in this process can be helped.
- If, else if and else structure is passed on to the students.
- Other steps should be done with the teacher and the teacher should transfer the logic of each step to the students.
- After completing the project, students can play the rock-paper-scissors game with PicoBricks for a short time.

## K-12 International Computer Science Standards

- 2-AP-10 6 - 8 Use flowcharts and/or pseudocode to address complex problems as algorithms.
- 3A-AP-13 9-10 Create prototypes that use algorithms to solve computational problems 3B-AP-10 11-12 by leveraging prior student knowledge and personal interests.
- 3B-AP-13 11-12 Use and adapt classic algorithms to solve computational problems.
- 3B-AP-13 11-12 Illustrate the flow of execution of a recursive algorithm.

wait until PicoBricks button

reset timer

for i in 10

# PART 7

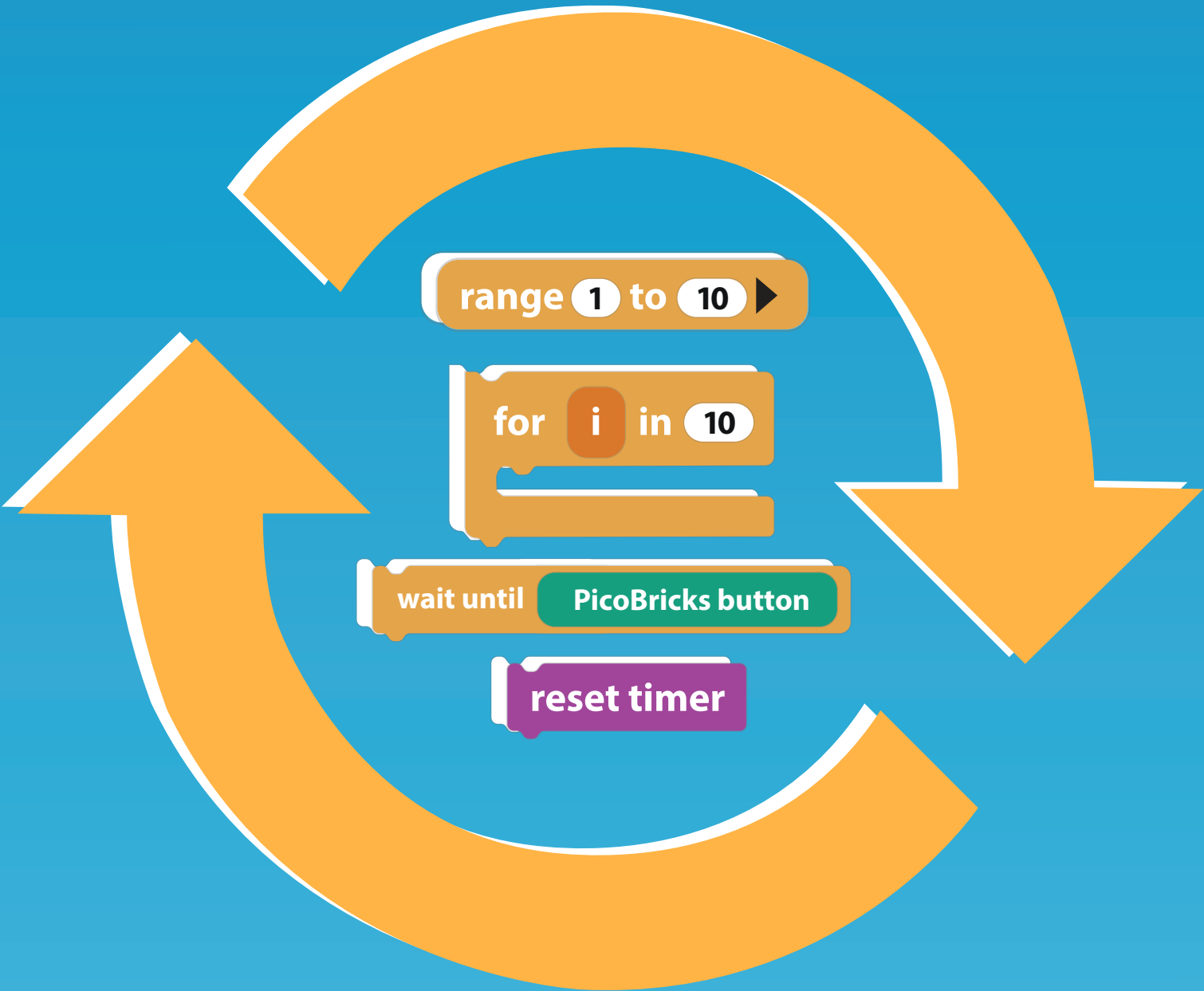
range 1 to 10

range 1 to 10

for i in 10

wait until PicoBricks button

reset timer



## What We'll Learn in This Part

In this part, students learn;

- The purpose of the for loop,
- Using the range statement with the for loop,
- Printing different types of expressions on the OLED screen with, the Join block,
- To use a timer and reset timer blocks.

## For Loop


In this part, we will use the for loop, which is one of the loop types. In the previous parts, we learned what loops are and what they are used for, and we developed a project by using the repeat loop. Let's start by understanding the difference between for and repeat loop.

When using the repeat loop, we were only specifying the ending value or the condition necessary for it to end inside the repeat loop. We had to set our starting value outside the loop.

In the for loop, we can define the end value of the loop inside the for block, without having to define our start value outside of the loop.

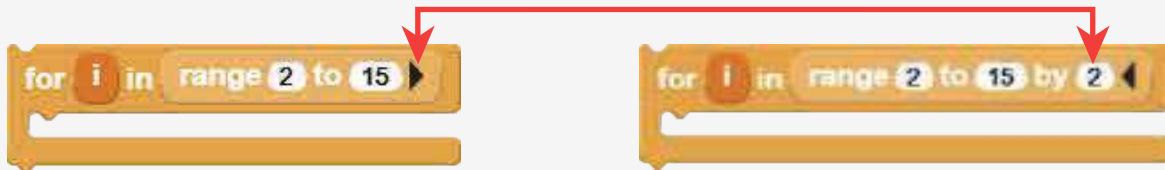
In the for loop, the variable "i" comes as defined, and the initial value of this variable i is "0". The end value of the loop is defined as "10", but we can change the ending value of the loop as we want.



Now let's get to know the  block used with the for a loop. Thanks to the "Range" block, we can determine the start, end and increment values of the for loop. The starting value of the for loop given below is "2" and the ending value is "15". The amount of increase is "1".



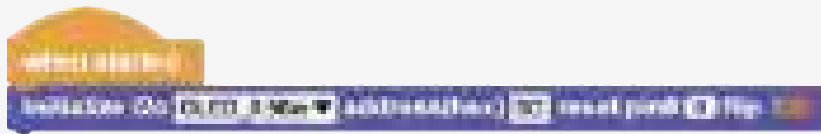
We can change the increment amount by clicking the arrow to the right of the "range" block. After clicking the arrow sign, we must enter the amount of increase in the window that opens. The for loop given below has a start value of 2, an end value of 15, and an increment of 2.



## The Counter with For Loop (1-10)

Now let's create code blocks that print the numbers 1 to 10 on the OLED screen using the for loop.

Since we will use the OLED screen, let's drag the "when started" block and the OLED screen startup block to our project page.



Let's drag the for loop to our project page. Since our starting value is "1" and our ending value is 10, we do not need to change these values because the starting value of the "i" variable is "1" and the end value is automatically set as "10" with the for loop.



The value we will print on the screen is the variable "i". Let's enter i value into the "write" block for this. Let's set the x and y coordinates as the middle of the screen (x=64 Y=32). Then, let's add the 500 milliseconds wait block to be able to observe the number changes on the screen. We can get the variable "i" by dragging it inside the a loop.





## The Counter with For Loop (10-1)

Now, let's reduce the numbers from 10 to 1 by 1 and print them on the screen. We can prepare this activity by making changes to the previous activity on our project page. First of all, we need to set our starting value as "10" since we will print backward from 10, but since we cannot change the starting amount of the "i" variable, let's drag the **range 1 to 10** block into the for loop, let's make the start value "10" and the end value "1". Then, let's wait for half a second and clean the screen each time in order to better follow the change on the OLED screen. We will use **clear** the block to clear the screen.



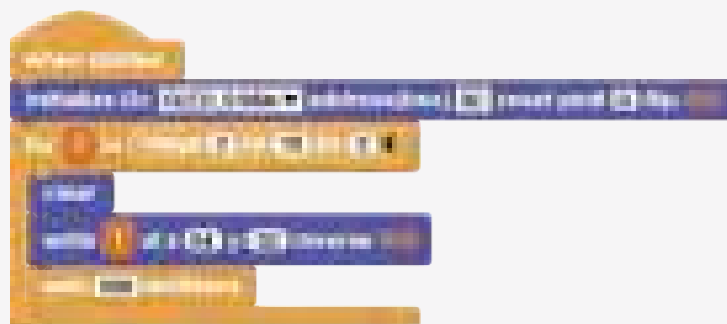
### A QUESTION

When the project is finished, where should we put **clear** the block to see the "1" value on the screen?

### ANSWER

## Counter Incrementing by 2 with For Loop (2-10)

In order to increase the numbers between 2 and 10 by 2 by 2, we must set the start value as 2, the ending value as 10, and the increment amount as 2. We can provide these values by entering the **range 2 to 10 by 2** block inside the for a loop. Then let's complete our code by specifying the expressions that will be written on the screen.

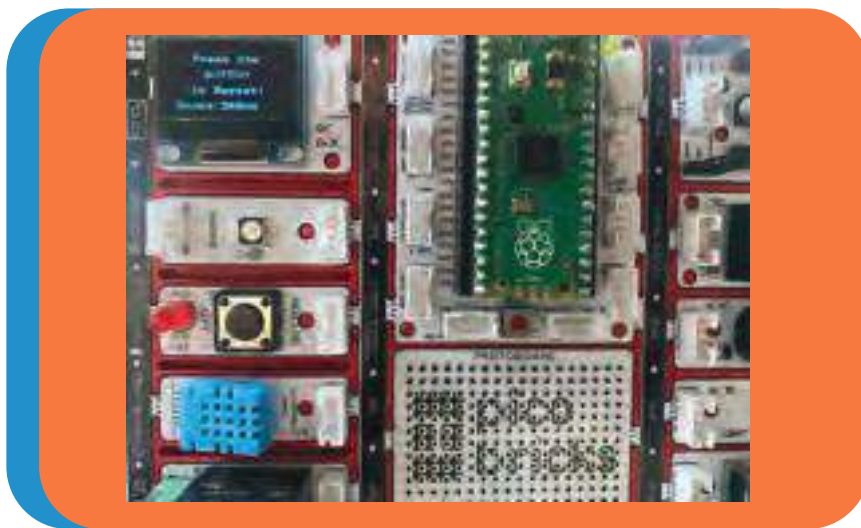


## Counters decreasing by 2 by 2 with For Loop

To print the numbers between 10 and 2, we can complete it by simply changing the range block. Our starting value will be “2”, our ending value will be “10” and our increased amount will be “2”. After creating `range 10 to 2 by 2` the block, drag it into the for block and create the necessary code blocks to print on the OLED screen. (You can copy these code blocks from the previous activity.)



## The Project of 7th Part: Show Your Reaction

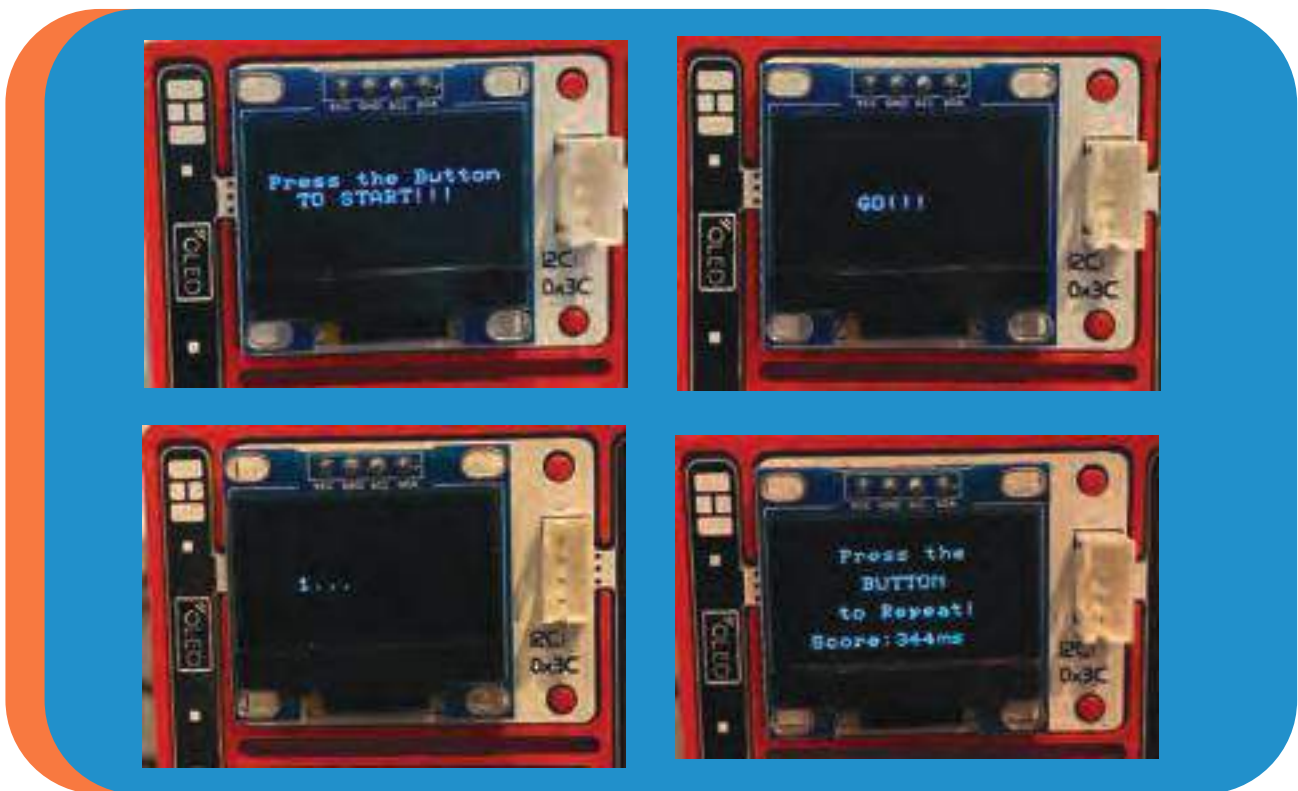


The Show Your Reaction project aims to measure the user's reflex. When the user presses the button, it gives a warning by turning on the red LED at a random time between 1 and 5 seconds after a waiting period of 3 seconds. After receiving this warning, the user is requested to react by pressing the button as soon as possible. After pressing the button, the response time is shown on the OLED screen and the buzzer works. Project Algorithm Start Wait until the button is pressed wait three seconds Turn on the red LED at random seconds from 1-5 and reset the time When the button is pressed again equate time to score. Write the score on the screen and start the buzzer. Return to item 1 when the button is pressed again.

## Project Algorithm

1. Start
2. Wait until the button is pressed
3. Wait three seconds
4. Turn on the red LED at random seconds from 1-5 and reset the time
5. When the button is pressed again equate time to score
6. Write the score on the screen and start the buzzer
7. Return to step 1 when the button is pressed again.

## Project Images



## The Video of The Project

For video narration of the project, you can visit [Picobricks.com](https://Picobricks.com) and [Robotistan INC YouTube channel](https://www.youtube.com/channel/UC...).



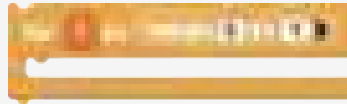
## MicroBlocks Code of The Project

Let's examine MicroBlocks code of the project.

When the project starts, "Press the button" and "TO START!" are displayed on the OLED screen. Create the following code blocks in your project page to print the expressions.





To write the code blocks that will run after pressing the button to create the block. Let's create the code block that shows the three seconds required for the game to start after the button is pressed, counting down from 3 on the screen. By using the information we learned in this section, we will create the for the block that decreases 1 by 1 from 3 backward on the screen. Using the range block, let's set the start value of the for loop as 3 and the ending value as 1 and the increment amount as 1. For this we need to create the block. Drag this range block we created into the for loop.

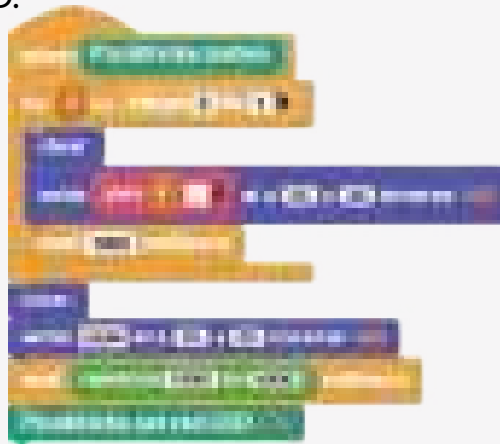




Since we will print "3...", "2...", "1..." on the screen at 1-second intervals by using the for loop, we first need to clear all the current statements on the screen. To do this, drag the 'clear' block into the for loop. We will use the "Write" block from the OLED Graphic library to print the "i" variable and the "..." expression on the screen. Until now, we have always printed an expression using the "Write" block on the OLED screen, but in this project, both the "i" variable and the "...". We want to print ".." statements. For this, we will use the 'data micro bricks' block from the "Data" blocks. On this block, let's drag i variable to the first field and write "..." in the second field, then drag it into the write block."


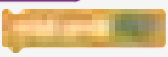
After creating the block, drag it into the for loop and drag the "wait" block immediately behind this block to wait a second.



After counting down from 3 on the screen and waiting for 3 seconds, we clean the screen with the "clear" block again and "GO!!!" on the screen. We print the expression. Let's create  blocks and drag them under the for a loop. After waiting for a random time between 1 and 5 seconds on the screen, create  block to turn on the red LED.




After these blocks, we want the button to be pressed and to print the time period from the moment the Red LED lights up to the moment it goes out after it is pressed. In MicroBlocks, there is a timer that records the moment we stop in the project since we start a program, and there is  the block in the "input" blocks for us to use this time period in our projects. We can reset the value of this timer block with  block.

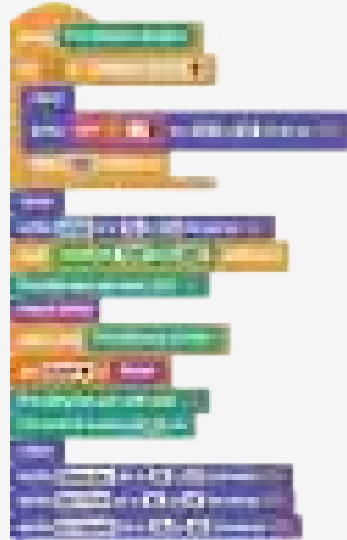
Now let's reset the time elapsed until then using  block and create code blocks that allow us to wait until the button is pressed.  the block is a conditional wait block. With the value we drag into it, it causes the program to stop until that condition is met. Since we want to wait until the button is pressed, let's drag the button block from the Picobricks library into it.

After creating the  block, drag it under the "reset timer" block.



Now let's create a score variable and assign the new value of the timer to this "score" variable. We will use the "score" variable later to print it on the screen.

Let's define the current time to the "score" variable by creating block  Let's start the buzzer after turning off the red LED. Then, let's print the expressions given in the picture below to the screen.

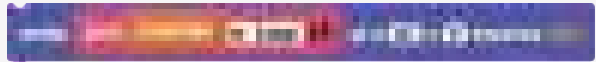


```

set score to timer
turn LED off
start buzzer
delay 1000 ms
print score

```

To complete our code, let's print the text "Score: ", the variable "score" and the text "ms" using the Join block.



```

join "Score: " score "ms"

```

Our code is ready. We can run it.



```

set score to timer
turn LED off
start buzzer
delay 1000 ms
print score
join "Score: " score "ms"

```

## K-12 International Computer Science Standards

- 2-AP-10 6-8 Use flowcharts and/or pseudocode to address complex problems as algorithms.
- 3A-AP-13 9 - 10 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
- 2-AP-16 6-8 Incorporate existing code, media, and libraries into original programs, and give attribution.
- 3A-AP-23 9-10 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.
- 3B-AP-22 11 - 12 Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
- 3B-AP-24 11-12 Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems.



# BONUS PROJECT



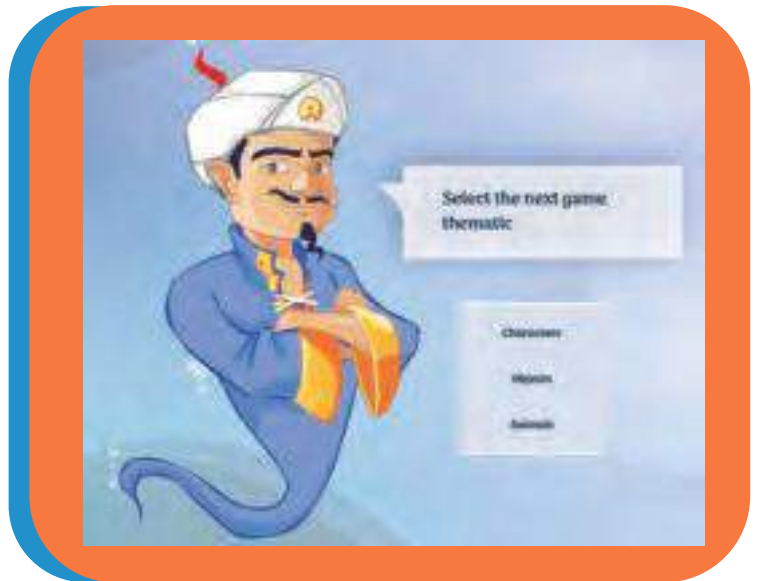
## Let's Play Akinator



Akinator is a game that allows us to identify the character in our minds with the questions it asks. This game determines the character you think by eliminating the questions it asks and the odds it hides in the database one by one. The Akinator game is constantly updating its database with the answers to the questions it asks. In this way, the game lays the foundations of artificial intelligence.

You can play Akinator game both on the mobile app and on the website.

Each student in the class play the Akinator game on their own computer a few times. The teacher can ask the students about a few of the following questions related to the game, so that the logic of the game can be better understood by the students.



### Some Advice For The Teacher

- How do you think Akinator can predict your character?
- What kind of logic might Akinator be choosing the questions he asks?
- If you were Akinator, what question would you ask yourself?

## The Number Guessing Game Details

The number guessing game project allows the PicoBricks to find a number that you keep in the range of 1-128 by directing the PicoBricks with the help of an IR sensor and remote control.

After pressing the button, PicoBricks makes the first guess and waits for the player to direct by using the "up", "down" and "ok" keys of the remote.

### Picobricks' prediction;

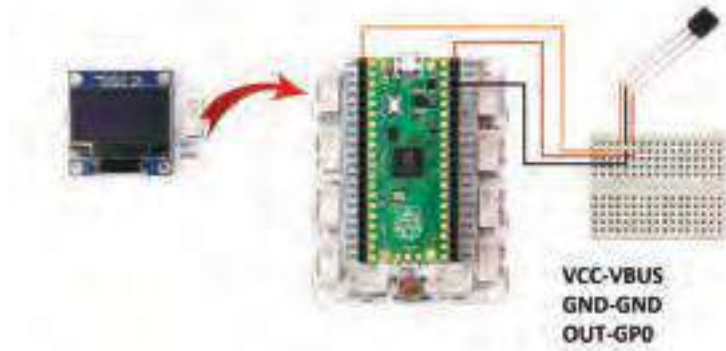


- if it is greater than the number you pick,
- if it is less than the number you pick,
- if PicoBricks guess your number correctly  
press it.

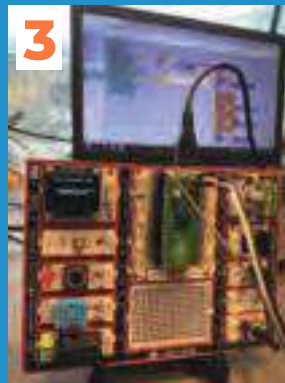
### Project Algorithm

1. Start when button is pressed.
2. Divide the max prediction value by .
3. If the player presses the up key, that is, if PicoBricks' prediction is less than the player's number, divide the guess interval by 2 again and add it to the current Picobricks' prediction.
4. If the player presses the down key, that is, if PicoBricks' prediction is greater than the player's number, divide the guess interval by 2 again and subtract it from the current PicoBricks' prediction.
5. If the player presses "ok" button, end the game and congratulate PicoBricks.
6. Repeat the 2nd, 3rd and 4th steps until the "ok" button is pressed.
7. Return the 1st step.

## Project Pin Diagram (PicoBricks v1.0)



## Project Images (PicoBricks v1.0)

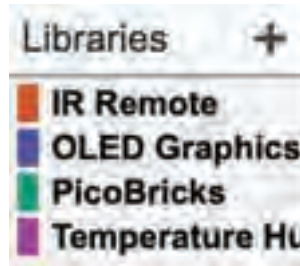


## Project Images (PicoBricks v1.2)



# MicroBlocks Code of The Project

Install the “IR Remote” library into the MicroBlocks online editor.



Firstly, when the project is started, print the expressions “Press the button” and “TO START” to on the OLED screen at the specified positions.




## When the button is pressed;

- Delete all the previous expressions on the OLED screen. Define a variable named “count” and set the initial value as 64.
- The “count” variable determines the range of numbers.
- Then, let’s define a variable named “guess” and set the initial value of this variable to the initial value of the “count” variable.
- By using “join” block, print the variable “My guess”, “:” and “guess” to the x and y coordinates specified on the OLED screen.
- Let’s print the expression “Press “OK” if I guess correctly” on the x and y coordinates on the OLED screen.




Now, that we've dragged the code blocks to our project page up to this point, we can move on to the loops.

Drag the “forever” loop onto your project page. All the operations we do after that will continue forever. Inside the “forever” block we will specify the condition statements. For this, let's first try to understand the communication between the IR sensor and the remote. Please, examine the table below. In our project, we will use 3 different keys on the remote. Each of these keys sends different values to the IR sensor, and we will use these values to determine the condition expressions.


The three different control keys below send three different values to block  .



- The down arrow key sends the value of 82.
- The up arrow key sends the value of 24.
- The OK key sends the value of 28.

We will use 3 different “if” blocks for the three keys we will use in the project. If the condition of the first of these “if” blocks comes to the value "24" with the help of the IR sensor, we will provide this condition expression by creating the block  . When this condition is provided, let's divide the "count" variable by and increase the “guess” variable by the count variable. Then let's print the “guess” variable to the screen.



Our condition statement in the second “if” block, if the value of “82” comes to the IR sensor with the help of the control, that is, if the player presses the down arrow key, we will provide this condition statement by creating the block . When this condition is provided, we will divide the variable “count” in half and subtract the variable “count” from the variable “guess”. Then, let’s print the guess variable to the screen.



Our third condition statement will be if the value of "28" comes to the infrared sensor with the help of the control, that is, if the "OK" button is pressed. When this condition is provided, We'll clear all the statements on the screen and print "Your number";":", PicoBricks' latest guess, and "Congratulations PicoBricks".



## The Project Is Ready!





# PART 8



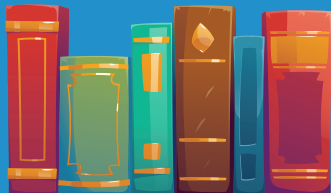


## What We'll Learn in This Part

- In this part, the student learns to;
- The purpose of the lists,
  - How to assign a list to a variable and use list blocks with MicroBlocks from PicoBricks programming tools.


# Lists

Lists are structures that allow us to use data in a more organized way. We can compare lists to a library. The number of rows and the number of books in the library determines the size of that library. Check out the sample library below.




There are 6 books in the library. If we think of this library as a list, this list is a 6-element list.



## Let's Define a List

Let's define a list with MicroBlocks now. For this, we will use the block  from Data blocks. We can adjust the list size by pressing the arrow key on the far right of the block. To give a name to the list you have created, you must first define a variable. Let's create a variable named "list". Let's create a list with 5 elements, let's determine the elements of this list as [a,b,c,d,e] and define this list to the "list" variable that we created earlier.



## Calling Element From The List

We can call an element in the list school we created using the block . When we first drag this block to our project page, it detects the text "Rosa" as a list and calls the 1st element in the list [R, o, s, a]. The output of this block is the letter "R".

Now let's call the 3rd element in the list we call "list" and print it on the OLED screen. Let's drag the list variable from the variables to the field that says "Rosa" in the block  and make the number of elements 3. We can call the last element and a random element by clicking the arrow button from where we change the number of elements.  Since we will call the 3rd element, let's write the

value "3" in this field with the keyboard.



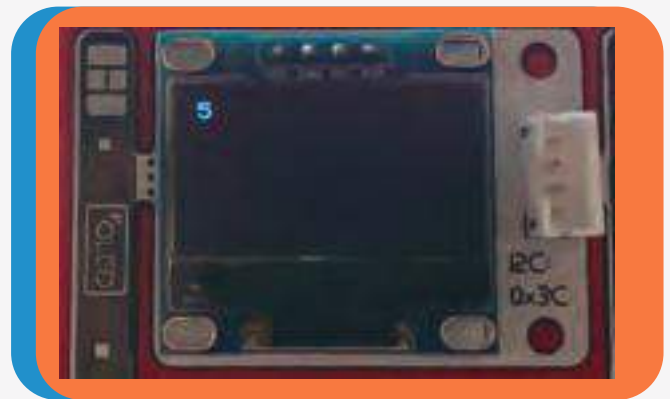
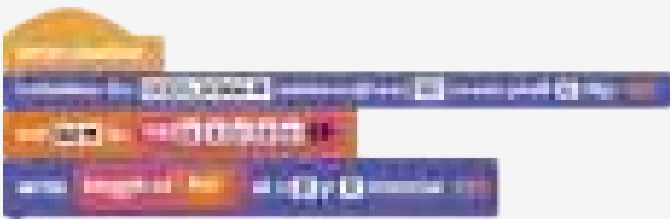
Let's create code blocks below to see this value on the OLED screen.



## Let's Find The Length of The List

We use the block `length of Rosa` to find the length of a list we have created. When we drag this block to our project page and click on it, we see that the block has the value 4 written on it because the [R,o,s,a] block has 4 elements.

Let's now print the number of elements of the list we created on the OLED screen, called "list", drag `length of Rosa` the block to the area where the text "Rosa" is located, create `length of list` block and print it on the OLED screen. Since the list we created consists of 5 elements, it will display 5 on the screen.


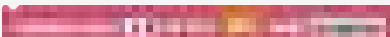


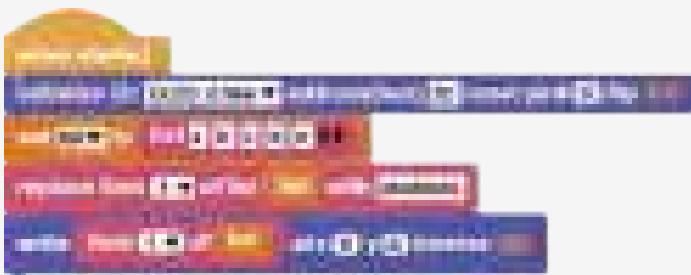
## Add Elements to The List

We use block `add fish to list` to add elements to a list we have created. Now let's add the element "f" to the list we have created called "list" and print it on the OLED screen. For this, let's create `add f to list list` the block and print it on the OLED screen.


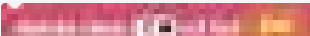


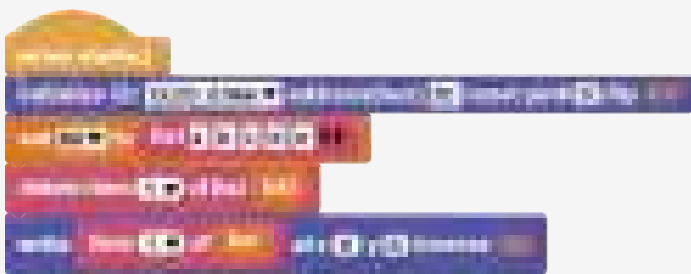
## Let's Change an Element in The List

We can change an element in the list we created using the  block. Now let's change the 1st element of the list we created called "list" to "picobricks" and print it on the OLED screen. For this, let's first create  the block, then create the following code blocks and print the 1st element on the OLED screen.



## Deleting an Element From The List

We use the  block to delete the element we want from a list we have created. Now let's delete the 1st element, "a", from the list we created and print the 1st element to the screen again. After using  block and deleting the 1st element, our 1st element will now be "b". Therefore, the letter "b" is written on the OLED screen.



## Let's Find an Element in The List

We use block **find a in cat** to find out which element of an element in the list we have created is in the list. Let's search for the "d" element with **find a in cat** block in the list we defined as "list". Let's print the rank of the element in the list on the OLED screen.

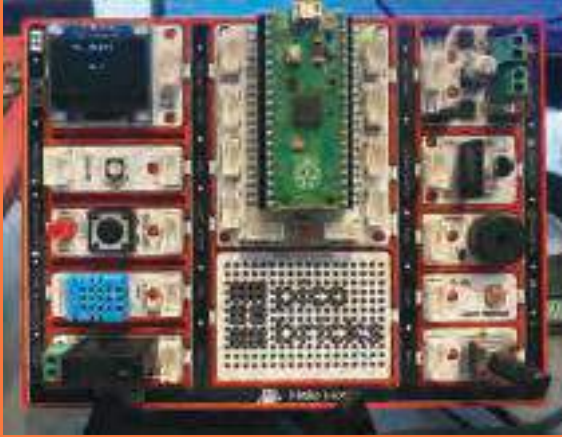


## Let's Call As Many Elements As We Have Specified Value Range

We use the **copy from** block to call the range of values we want in a list we have created. For example, when you drag this block to our project page for the first time, it perceives the word "smiles" as a list and calls it from the 2nd element to the last element, so when we click on it, we see the "miles" value. In the "smiles" list, we can enter the value range by clicking the arrow at the end of the block to call the values starting from the 2nd element up to the 5th element. In this block, **copy from** it calls the elements from the 2nd element to the 5th element and its output is "mile". Now, let's call only the "b, c, d" elements from the list called "list" that we created before, we will use **copy list from 2 to 4** block for this.



## The Project of The 8th Part: First 100 Digits of Pi Number



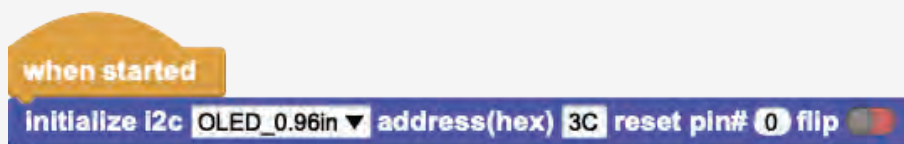
In this project, we will print the first 100 digits of Pi to the screen in order. While writing the digits on the screen, the number of digits on the screen will be indicated on the screen at the same time. This project can be used as a small exercise project designed to memorize the digits of pi.



### Project Algorithm

1. Start
2. Except for the first digit of the pi number, that is, the other 99 digits after the comma, create a variable named pi and define it to the variable pi.
3. Define a "counter" variable to determine which digit it is in and define its initial value as "1".
4. Define a for loop that will return as many digits of Pi as the variable "Pi".
5. Each time the for loop is repeated, increase the "Counter" variable by 1 and print the "counter" variable with the expressions "th" and "digit" on the OLED screen.
6. Let's print the number of pi with its digits, namely the variable pi, with the expressions "3" and "," on the screen.
7. Wait for half a second to print the digits at half-second intervals.
8. Return to the first step.

## Let's Create MicroBlocks Code of The Project

Let's drag the "when started" block to our project page and start the OLED screen as we will print the steps on the OLED screen.




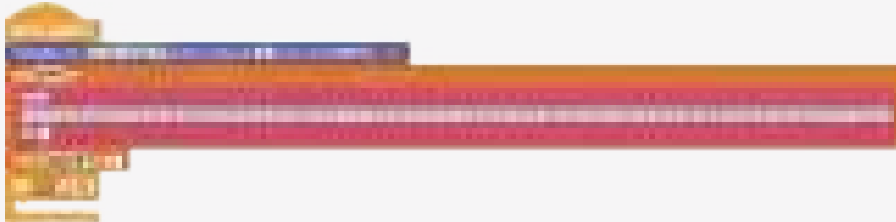
Let's create a variable named "pi". We will define the list with 99 digits of the number pi except for the first digit in this variable. Instead of entering the digits of the number one by one, we will use  block to provide a comma between each digit. When we put a comma between the values we entered in  block, we perceive the value between the two commas as the element of the list. Let's create




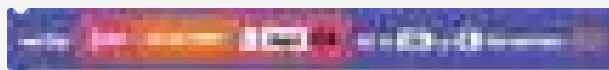
block by writing a comma between the 99 elements that come after the comma in the block and define it to the variable pi. Then let's create another variable called counter and initialize it to "1".




Now we can move on to the loop phases we use in our project. After dragging the for loop block into our project, drag  block into the for loop block to determine the number of iterations of the loop and create the for block.



Let's create statements that will repeat in the loop. Let's increase the "counter" variable we defined by "1" to increase the digit value each time the loop repeats. For this, create block . To print the digit value to the screen, use the join block to print the "counter" variable, "." and "digit" expressions on the OLED screen.



Create block  to print the current digit of pi after the digit value. Thanks to this block, each time the loop repeats, the elements in the pi list will be displayed in order. Let's drag the half-second wait block to print each stepper at half-second intervals and run our project.

### Let's Run The Code of The Project





## K-12 International Computer Science Standards

- 3B-AP-20 11 -12 Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (code documentation) in a group software project.
- 3A-AP-14 9-10 Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.
- 3A-AP-13 9-10 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
- 2-AP-11 6- 8 Create clearly named variables that represent different data types and perform operations on their values.
- 1B-AP-10 3-5 Create programs that include sequences, events, loops, and conditionals.
- 1B-AP-09 3-5 Create programs that use variables to store and modify data.



command ▶

reporter ▶

command ▶

reporter ▶

reporter ▶

# PART 9

define command ▶

define reporter ▶



## What We'll Learn in This Part

In this part, the student learns to;

- Understand for what purpose "My blocks" blocks are used,
- Create your own blocks using the define block,
- Explain the difference between Command and Reporter blocks.

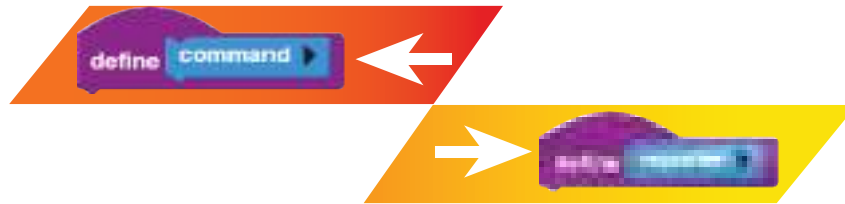
## Define - Creating Your Own Block

Thanks to “define”, we can create our own commands in programming environments. For example, in a project, we are asked to constantly detect prime numbers and perform operations on those prime numbers. So, instead of constantly detecting prime numbers, we can define a function where we find prime numbers and only call that function. This application also makes the code we write more simple and understandable.

While developing any projects in MicroBlocks IDE, we can create our own code blocks and assign tasks to these blocks. We do this using define blocks. “My Blocks” menu is used to access MicroBlocks define blocks. In the "My Blocks" menu, there are two options called command block and reporter block to define your own block. Using these options, we can create our own blocks and assign tasks to these blocks.



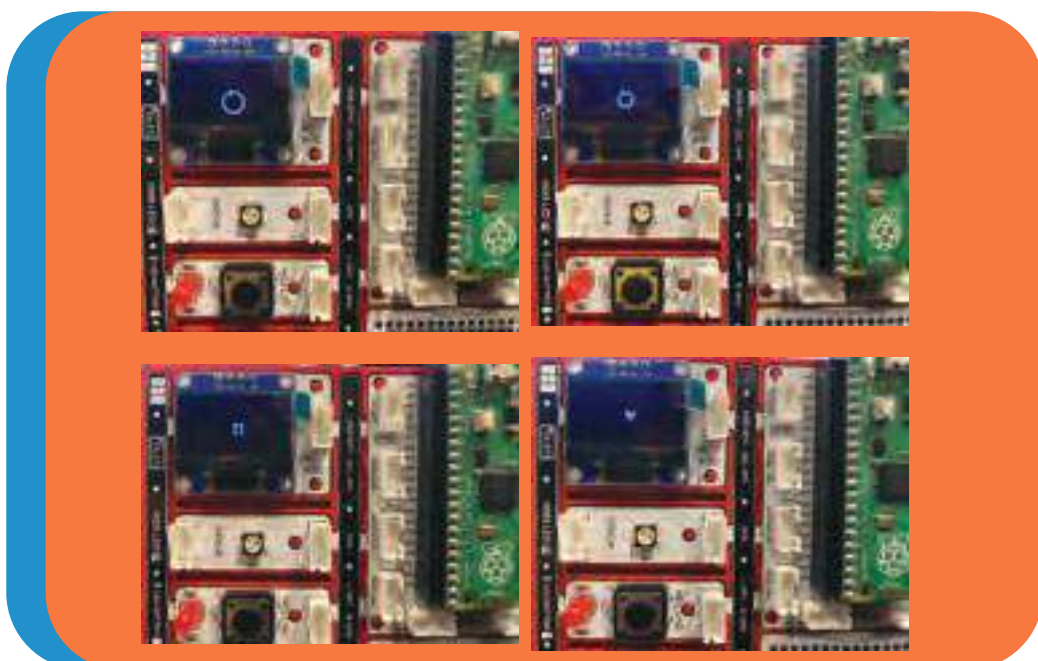
## The Difference Between Command and Reporter



Command and reporter blocks are the types of blocks that we use for different purposes, and we can easily understand this difference from the structure of the blocks. When we look at the structure of `command` the block, we can see that it is like a puzzle piece, which means we can drag the command blocks we created under the other code blocks. When we look at the structure of the block `reporter`, it is an oval structure, so we can understand that it is one of the code blocks that we can use in condition expressions etc.. Now, we can better understand the difference between these two blocks by making a few activities.

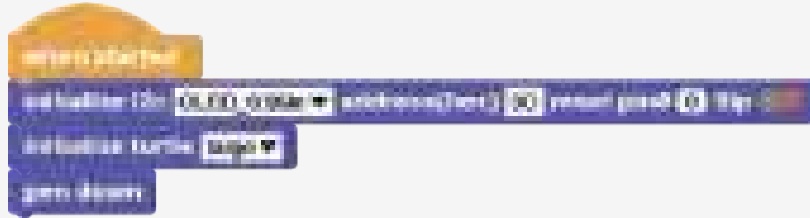
## The Shapes with Define

In this project, we will transform the square, triangle, hexagon, and octagon shapes we created using the turtle library into a single block using the `define(command)` block in the "my blocks" tab. Now, when we call that block, we will print the shape we want on the OLED screen.



## MicroBlocks Code of The Project


Let's install the OLED and Turtle libraries that we will use in our project to MicroBlocks IDE. Then let's call our project OLED screen and turtle cursor. Then, let's drag the block **pen down** to our project page to track the path of the turtle cursor on the OLED screen.



Now, let's create code blocks that create octagon, hexagon, square and triangle shapes by using the "My Blocks" tab.



Let's create a command block called "octagonal" by following the above steps. Thanks to this block, we will create an octagon shape on the OLED screen with the turtle cursor.

When we follow the above steps, you will observe that block  is formed on our project page. To create the octagon shape on the OLED screen with the Turtle library, the cursor must move 10 steps forward after turning 45 degrees eight times. For this, we should create the following block structure.



Let's create a hexagon block. To create the hexagon shape on the OLED screen with the Turtle cursor, the cursor must move 10 steps forward after turning 60 degrees six times. Let's define these operations in the defined block we have created called "hexagonal" by creating the block structure below.



Let's create a square block. To create the square shape on the OLED screen with the Turtle cursor, the cursor must move 10 steps forward after turning 90 degrees four times. Let's define these steps in the command block we created named "square" as in the code blocks below.



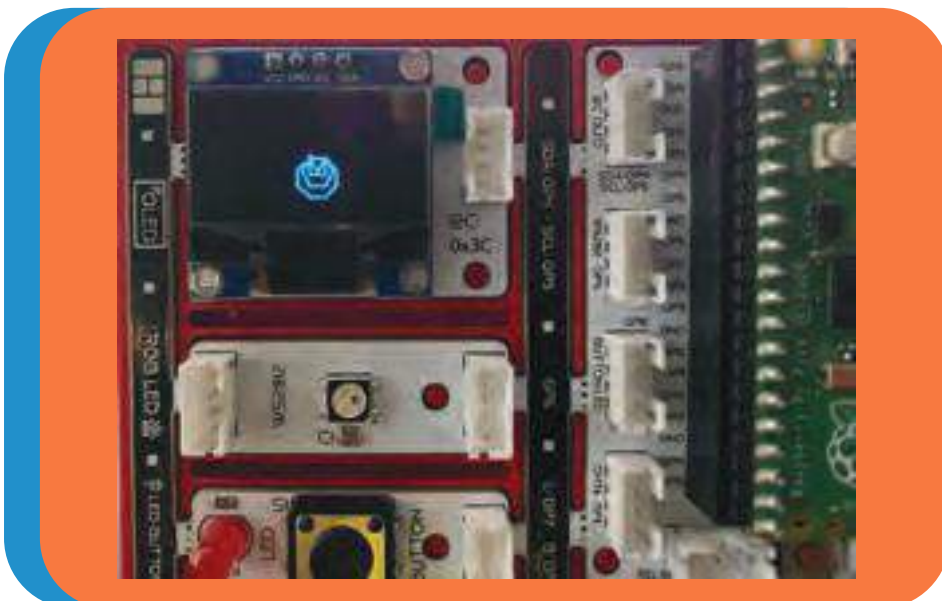
Let's create a triangle block. To create a triangle shape on the OLED screen with the Turtle cursor, the cursor must move 10 steps forward after turning 120 degrees three times. For this, let's list these steps in the command block we created called "triangle" as in the code blocks below.



Let's call these code blocks we created one by one to our project and run them. After observing the shapes formed on the screen, let's complete our project by calling all the shapes one after another.



## MicroBlocks Code of The Project Is Ready



## Double The Number with Define (Reporter)

In this project, we will print twice the number entered on the OLED screen by using the "reporter" structure from define blocks.



### Project Algorithm

1. Start
2. Define a function called "double"
3. Define a variable called "number" inside the "double" function
4. Run the "number" variable by constantly multiplying it by 2
5. Call the "double" function on the OLED display for the specified value
6. Return to the first step



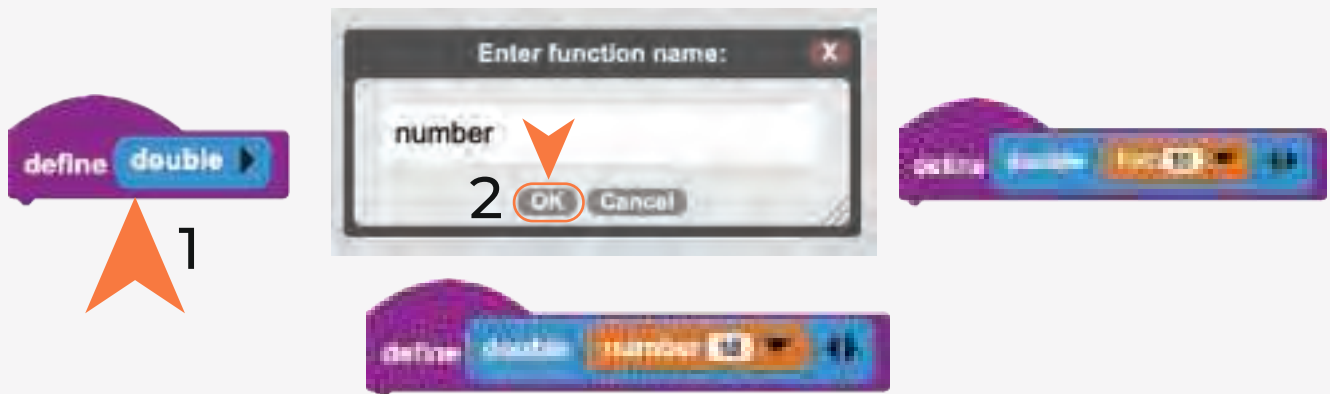
## MicroBlocks Code of The Project

First of all, let's prepare our defined function. Follow the steps below to create a reporter block called "double" by entering the "My Blocks" tab.



Let's follow the steps below to create a number variable in the double block we created by following the steps above.

To create the "number" variable in the "double" function created on our MicroBlocks IDE editor page, let's click on the arrow icon above the "double" block as follows.



Let's define the "number" variable as described above. Let's run double the "number" variable each time we call the "double" block, by using the `return 0` block from the "Control" blocks. Let's create the `return number` block and drag it under the "double" function, to do this.



Now, let's print the value given by the "double" function on the OLED display module. Let's create the following code blocks to print the value given by the `double 10` block on the OLED display from the "My Blocks" tab.





## MicroBlocks Code of The Project Is Ready



## K-12 International Computer Science Standards

- 3A-AP-13 - 9-10 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
- 2-AP-19 -6-8 Document programs to make them easier to follow, test, and debug.
- 3A-AP-15-9-10 Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.
- 3A-AP-21 - 9-10 Evaluate and refine computational artifacts to make them more usable and accessible.
- 2-AP-16- 6-8 Incorporate existing code, media, and libraries into original programs, and give attribution.
- Use flowcharts and/or pseudocode to address complex problems as algorithms.



**STEM 1**

# STEM Project 1 - Action Reaction

As Newton explained in his laws of motion, a reaction occurs against every action. Electronic systems receive commands from users and perform their tasks. Usually, a keypad, touch screen, or button is used for this job. Electronic devices respond verbally, in writing, or visually to inform the user that their task is over and what is going on during the task. In addition to informing the user of these reactions, it can help to understand where the fault may be in a possible malfunction.

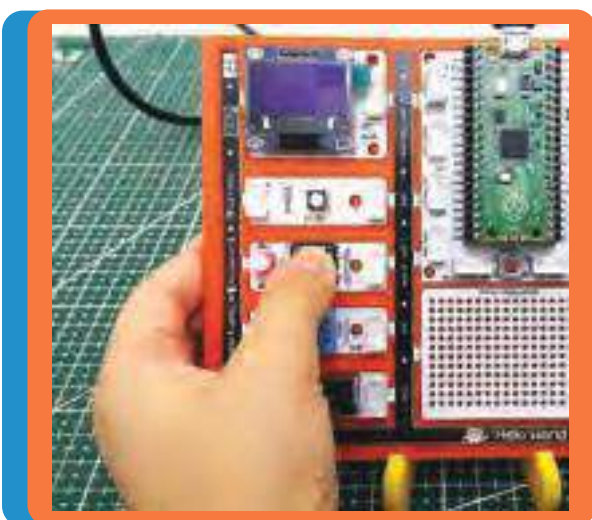
## Project Details

Different types of buttons are used in electronic systems. Locked buttons, push buttons, switched buttons... There is 1 push button on Picobricks. They work like a switch, they conduct current when pressed and do not conduct current when released. In the project, we will understand the pressing status by checking whether the button conducts current or not. If it is pressed, it will light the LED, if it is not pressed, we will turn off the LED.






## Project Algorithm

1. Start
2. If the button is pressed, turn on the LED module.
3. Turn off the LED when the button is not pressed.
4. Return to the first step


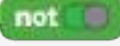





## MicroBlocks Code of The Project

Let's install the PicoBricks library on our project page. Then, let's drag the  block from the "control" block to our project page and drag the "PicoBricks button"  block from the PicoBricks library into it. Thanks to this block we have created, all the code blocks we drag under will be realized when we press the button on PicoBricks. Since we want the LED module to work when we press the button, let's drag the  block to our project page and create the code block below.

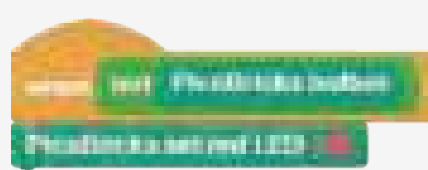


Now, let's create the code blocks necessary to create the expressions that will happen when the button on PicoBricks is not pressed. Firstly, let's drag the  block to our project page. Then, let's drag the  block from the operator blocks into this block. The expression we drag into the "note" block corresponds to the situations in which that block is negative. When we drag the "PicoBricks button" block into the "note" block, the code block is ready to detect the situations in which the button is not pressed. Then, let's drag these blocks into the "when" block.

Every block we drag under this block  works when the button is not pressed. Since we want the LED to turn off when we do not press the button, let's create the following code blocks on our project page.



## The Code of The Project Is Ready!



# STEM 2



## STEM Project 2 - Dominate The Rhythm



Many events in our lives have been digitized. One of them is sounds. The tone and intensity of the sound can be processed electrically. So we can extract notes electronically. The smallest unit of sounds that make up music is called a note. Each note has a frequency and intensity. With the codes we will write, we can adjust which note should be played and how long it should last by applying frequency and intensity.

In this project, we will prepare a music system that will play the melody of a song using the buzzer module and adjust the rhythm with the potentiometer module with Picobricks. You will also learn the use of ariables, which has an important place in programming terminology, in this project.

### Project Details

Potentiometer is analog input module. It is variable resistance. As the amount of current flowing through it is turned, it increases and decreases like opening and closing a faucet. We will adjust the speed of the song by controlling this amount of current with codes. Buzzers change the sound levels according to the intensity of the current passing over them, and the sound tones according to the voltage frequency. With MicroBlocks, we can easily code the notes we want from the buzzer module by adjusting their tones and durations.

#### Project Algorithm

1. Start
2. Assign the value of the potentiometer to a variable and write it to the display.
3. Wait until the button is pressed
4. If the button is pressed, play the determined tones at the specified speed according to the value of the potentiometer.
5. Repeat the second step twice.
6. Apply the second step until the button is pressed again.

## MicroBlocks Code of The Project

- 1) Since we will print the speed value from the potentiometer on the OLED display when the project starts, let's start the OLED display and print "Speed:" on the display.
- 2) Let's assign the value from the potentiometer to a variable named "rhythm". The value of the potentiometer is between 0-1023. We will use the block to change this value range to 1-7. This block is included in the operator blocks, but it is an advanced block. After clicking "show advanced blocks", it will appear between operator blocks.



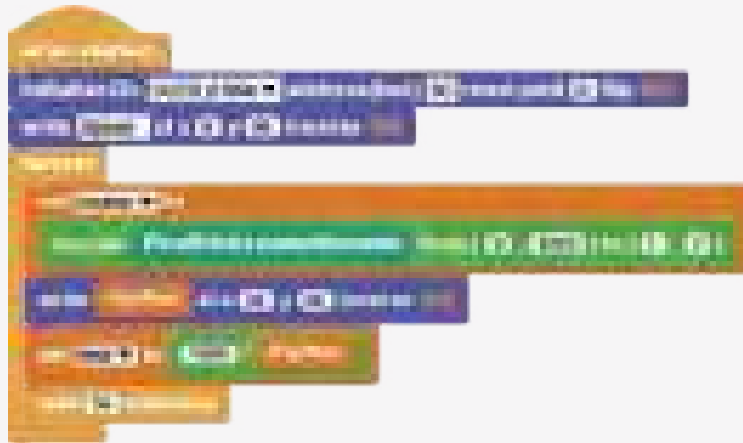
- 3) Let's create the following code blocks to reduce the value of the potentiometer between 1-7 and define it to a variable called "rhythm".



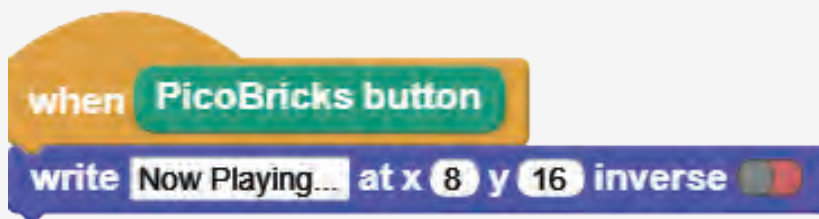
- 4) Let's print this variable on the OLED display. Then, let's create another variable called "beat" and define this value as 1000 / "rhythm" variable. This action determines how many seconds the tones will sound. Wait 50 milliseconds after this process.



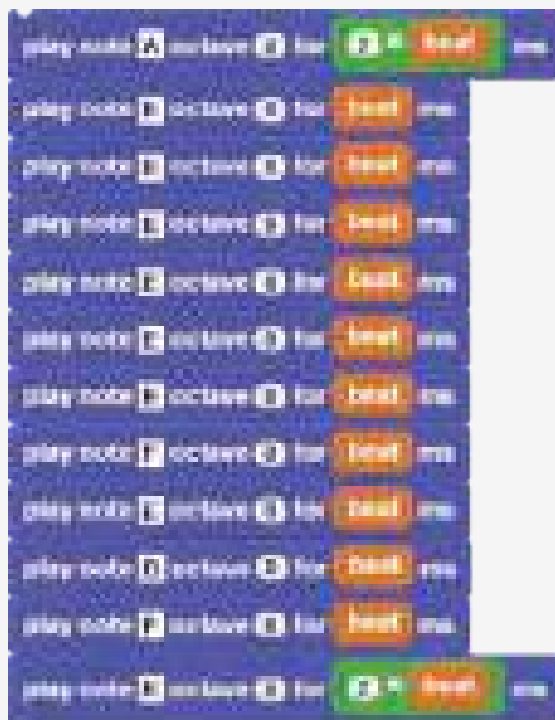
5) Since I want these steps to continue continuously, let's drag it into the "forever" block.



After pressing the button, let's do the following steps to make the tones play at the speed we have determined.

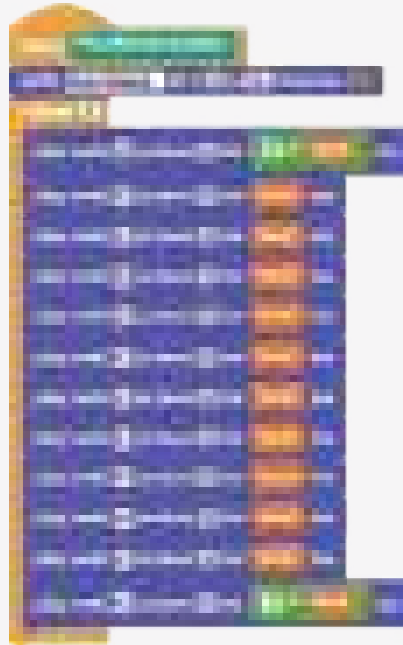


5) Our code is ready. We can run it.

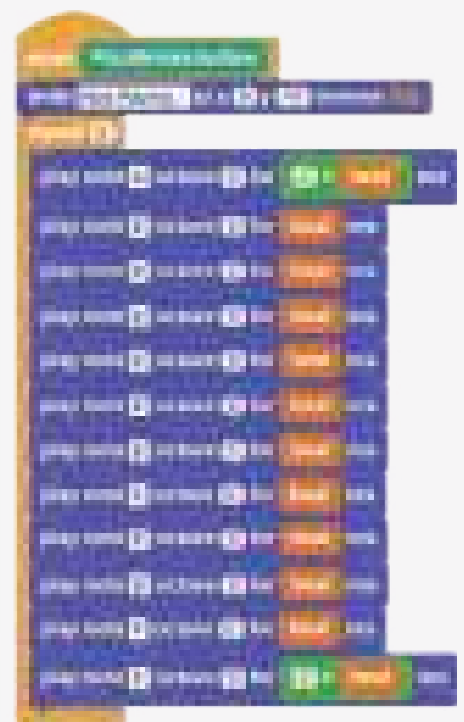
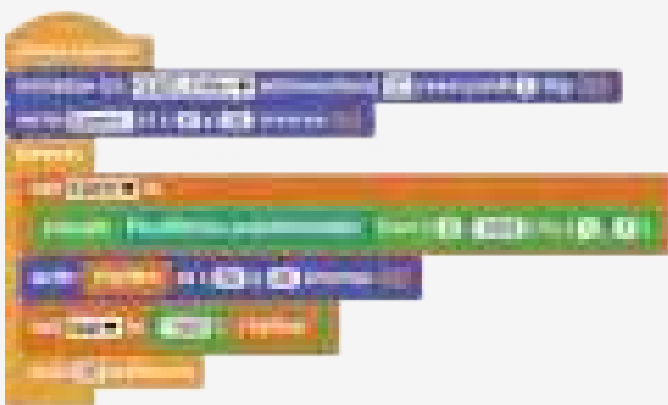




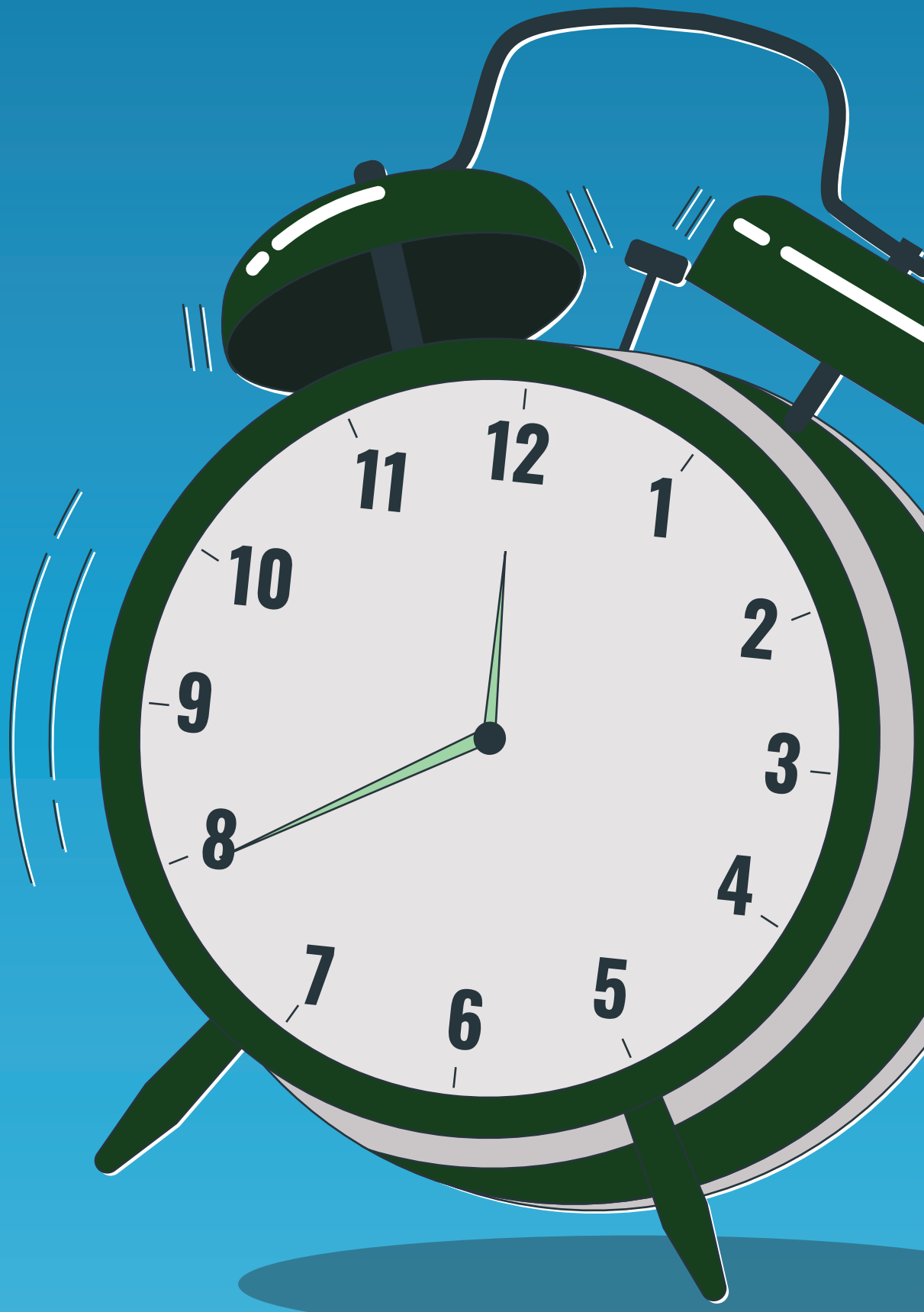
- 4) Since we want to create a melody by repeating these tones twice, let's take them into the "repeat" block and set the repeat value to 2.



## The Code of The Project Is Ready!



# STEM 3



## STEM Project 3 - Alarm Clock



Global warming is affecting the climate of our world worse every day. Countries take many precautions and sign agreements to reduce the effects of global warming. The use of renewable energy sources and the efficient use of energy is an issue that needs attention everywhere, from factories to our rooms. Many reasons such as keeping road and park lighting on in cities due to human error, and the use of high energy-consuming lighting tools reduce energy efficiency. Many electronic and digital systems are developed and programmed by engineers to measure the light, temperature and humidity values of the environment and ensure that they are used only when needed and in the right amounts.

In this project, we will create a timer alarm that adjusts for daylight using the light sensor in PicoBricks.

### Project Details

In this project, we will make a simple alarm application. The alarm system we will design is designed to sound automatically in the morning. For this, we will use an LDR sensor in the project. At night, the OLED screen will display a good night message to the user, in the morning, an alarm will sound with a buzzer sound, a good morning message will be displayed on the screen, and the RGB LED will light up in white for light notification. The user will have to press the button of Picobricks to stop the alarm. After these processes, which continue until the alarm is stopped, when the button is pressed, the buzzer and RGB LED will turn off and a good day message will be displayed on the OLED screen.



## Project Algorithm

1. Start
2. Print "Good Night" on the OLED screen.
3. If the PicoBricks LDR module has a value greater than 90 ( when the sensor detects the daylight), wait three seconds.
4. Until the button is pressed, print "Good morning" on the OLED screen, turn on RGB LED and buzzer.
5. If the button is pressed, print "Have a nice day" on the OLED screen and turn of RGB LED.
6. Return to the first step.

## MicroBlocks Code of The Project

When our project code start, let's create the following code blocks to wait 2 seconds after printing "Good night" on the OLED screen.



When the LDR module detects the daylight, let's create the necessary block to start the code blocks we want to run.

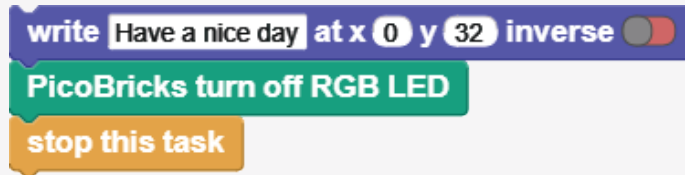
Let's assume that the value of the LDR module is greater than 90 when the sun is up (This value differs from environment to environment.). When this condition is happened, let's create the following code blocks to run the desired code blocks.



When the sun is up, that is, when the value of the LDR sensor is greater than 90, wait 3 seconds and print "Good morning" on the OLED screen until the button is pressed, turn on the RGB LED and activate the buzzer. Let's drag the following code blocks to perform these operations.



After pressing the button, let's print "Have a nice day" on the OLED screen, turn off the RGB LED and stop the running blocks. Let's drag the following code blocks to run these operations.



```
write "Have a nice day" at x 0 y 32 inverse
PicoBricks turn off RGB LED
stop this task
```

## The Code of The Project Is Ready!



The background is a solid green color with a subtle gradient. It is decorated with several horizontal strings of colorful lights. The lights are in various colors including red, yellow, cyan, and white. The strings are arranged in a way that they appear to be hanging from the top and curving downwards, creating a festive and celebratory atmosphere.

# STEM 4

## STEM Project 4 - Know Your Color



LEDs are often used on electronic systems. Each button can have small LEDs next to each option. By making a single LED light up in different colors, it is possible to do the work of more than one LED with a single LED. LEDs working in this type are called RGB LEDs. It takes its name from the initials of the colors names Red, Green, and Blue. Another advantage of this LED is that it can light up in mixtures of 3 primary colors. Purple, turquoise, orange...

In this project, you will learn about the randomness used in every programming language. We will prepare an enjoyable game with the RGB LED, OLED display, and button module of PicoBricks.

### Project Details

The game we will build in the project will be built on the user knowing the colors correctly or incorrectly. One of the colors red, green, blue, and white will light up randomly on the RGB LED on PicoBricks, and the name of one of these four colors will be written randomly on the OLED screen at the same time. The user must press the button of PicoBricks within 1.5 seconds to use the right of reply. The game will be repeated 10 times, each repetition will get 10 points if the user presses the button when the colors match, or if the user does not press the button when they do not match. If the user presses the button even though the colors do not match, he will lose 10 points. After ten repetitions, the user's score will be displayed on the OLED screen. If the user wishes, he may not use his right to reply by not pressing the button.





## Project Algorithm

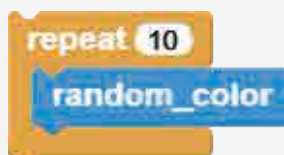
1. Start
2. Print a random color from green-blue-red-white on the OLED display.
3. Activate a random color among the green-blue-red-white colors on the RGB LED.
4. If the color on the OLED display and the color on the RGB LED module are the same and the button is pressed, increase the score by 10.
5. The color on the OLED display is the same as the RGB LED, but if the button is not pressed, decrease the score by 10.
6. If the color on the OLED display and the RGB LED are not the same, leave the score the same.
7. After repeating these operations 10 times, print the score on the display.
8. Return to the first step

- 1) Since we will be using the OLED display module in our project, let's start by initializing the OLED display.
- 2) Let's turn off the RGB LED at the start of the project to reset the RGB LED module every time the project starts.
- 3) Let's create a variable named "score" to record our score throughout the game.
- 4) After printing "The game begins" on the OLED display, wait 2 seconds and clean the OLED display.

Let's create the following code blocks on our project page to provide the above mentioned items.



- 5) Just below these blocks, the main operations necessary for our game will be done. Since the game will repeat 10 times, let's drag the "repeat loop" and set its value to 10.





Let's analyze "random\_color" blocks through the following blocks.



Inside this block, there will be a repeat cycle that repeats 10 times. In this Loop; We will light one of the 4 colors randomly in the RGB LED module and in the same way, we will print one of these 4 colors randomly on the OLED display;

- Let's define a variable called "randomColorIdx" and this variable will allow us to randomly flash one of the 4 colors on the RGB LED module. Let's create the block that makes the value of this variable a random number between 1 and 4.
- Let's create another variable named "randomColorNameIdx". This variable will allow us to randomly print 1 of the 4 colors determined on the OLED display. Let's assign a random number between 1 and 4 to this variable.
- Let's create another variable named "colorName", let's create a list with 4 elements and make the value of this variable the name of the 4 color options that we specify the elements of the list. Then, according to the value of the variable we created in the previous article, let's assign an element of this list to the variable named "colorName".
- In the same way, let's create another variable named "RGB\_color" and the list we will define will determine the value of this variable. Let's define a list with 4 elements and make the list 4 RGB color values whose elements we have determined. Let's determine the value of the "RGB\_color" variable according to the value of the "randomColorIdx" variable we defined earlier.
- Let's set the value of the PicoBricks RGB LED Module to the "RGB\_color" variable.
- Let's make the "colorName" variable for the expression that will be written on the OLED display.
- After all these operations, let's add a 50-milliseconds wait block.

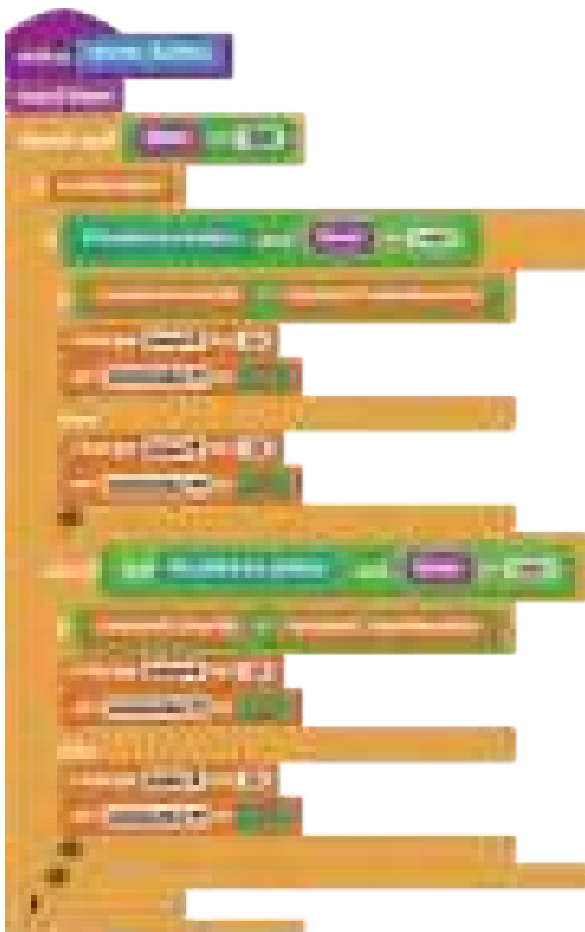
- 7) Let's define another variable called "noSelection" and drag the value of this variable from the operator blocks to the key block. Let's set the value of this key explicitly. This variable will allow us to detect whether the button has been pressed or not.



- 8) Let's create another "define" block named "check\_Button" to detect if the button has been pressed.



Let's examine the "check\_Button" blocks through the Blocks below



- There is a 1.5 second time limit for the player to react in the game. To start the time, it is necessary to reset the "timer" firstly. For this, let's drag the "reset timer" block.

- To understand that the player pressed the button within 1.5 seconds, the player must repeat the cycle until the timer is greater than 1.5 seconds.

- In this "repeat until" block, we check with the "noSelection" variable if the player did not pressed the button. If "no selection" is in the public key value as we first defined;

- We check if the button is pressed within 1.5 seconds. If the button is pressed within 1.5 seconds and the color written on the display and the color flashes in RGB LED are the same, we increase the "score" variable by 10 and turn off the "no selection" switch. If not, we reduce the "score" variable by 10 and turn off the "no selection" switch.

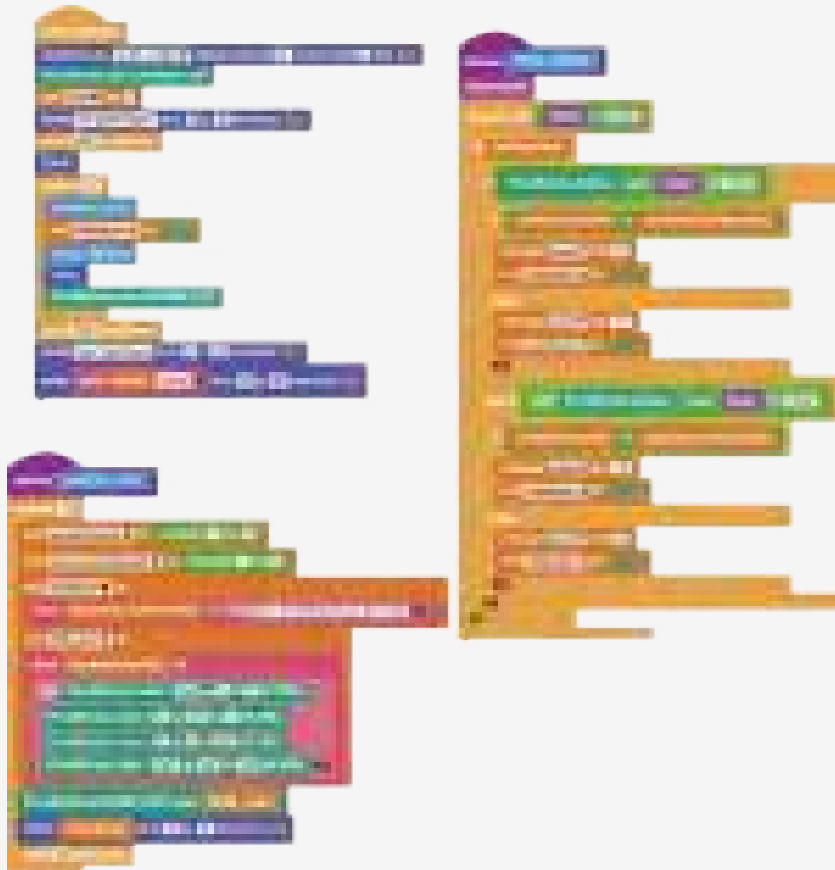
- If the button is not pressed and the time exceeds 1.5 seconds, at the same time, if the color written on the display and the color flashes in RGB LED are the same, we decrease the "score" variable by 10 and turn off the "no selection" key. If not, we increase the "score" variable by 10 and turn off the "no selection" key.

9) After the “check\_Button” block works, we clear the display and turn off the RGB LED module.

10) Finally, we print the “score” variable to the display.



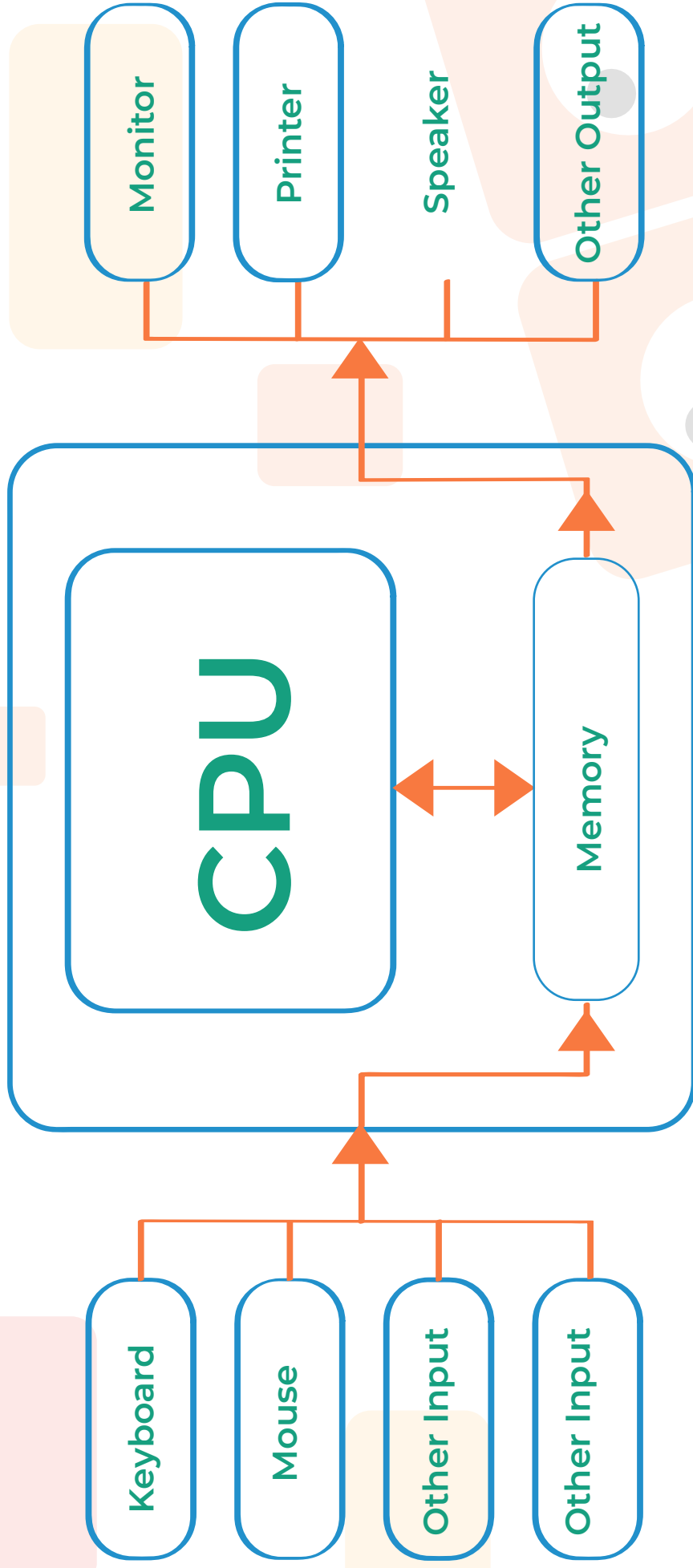
## The Code of The Project Is Ready!



# **Print and Distribute to Class**



## **Fascicle and Worksheet**



**1. Processor:** Small chips on the motherboard are called processors. We can think of the processor as the brain of the computer. The faster the processor, the faster your computer.

**2. Storage Units:** Computers store information using storage units. Storage units are divided into short-term and long-term.

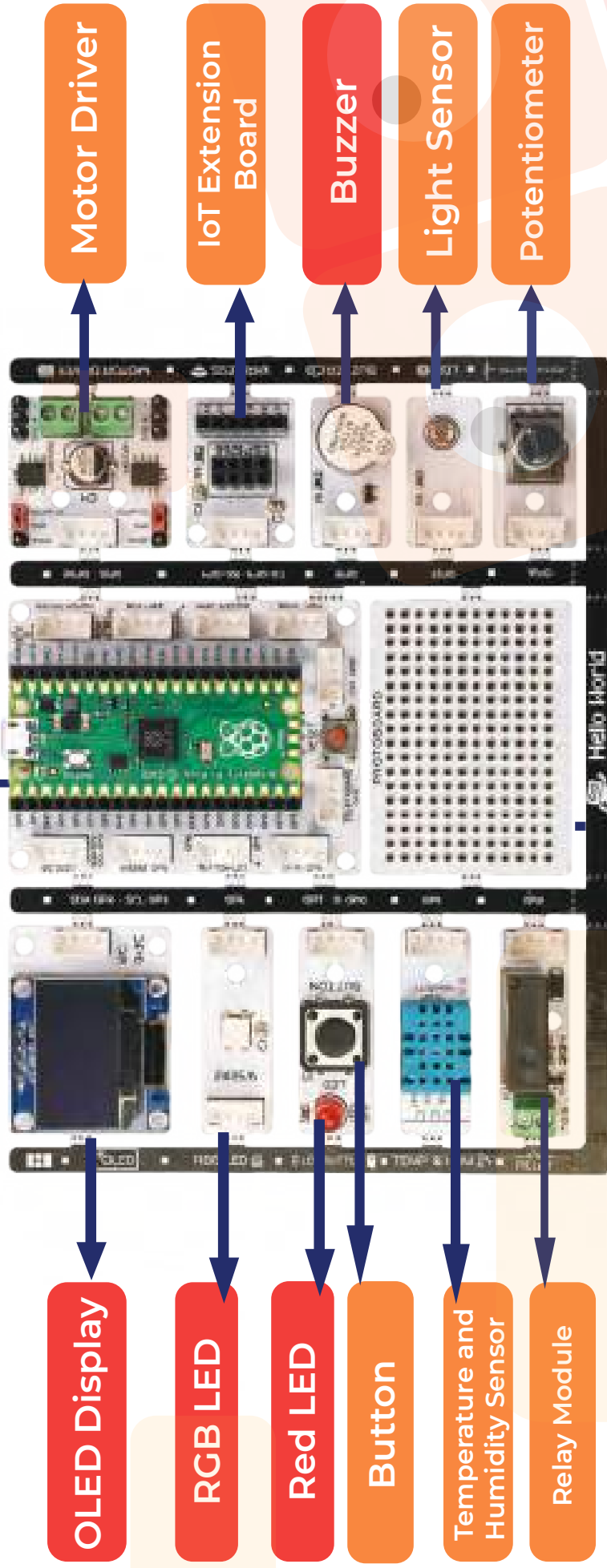
a) **Short-Term Memory:** RAM (Random Access Memory) is the place where information transmitted to the computer is temporarily stored.

b) **Long-Term Memory:** LTM (Long-Term Memory) is a memory unit where information is permanently stored, not deleted.

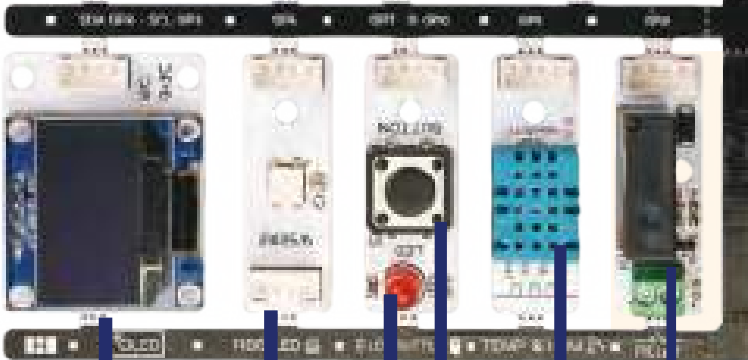
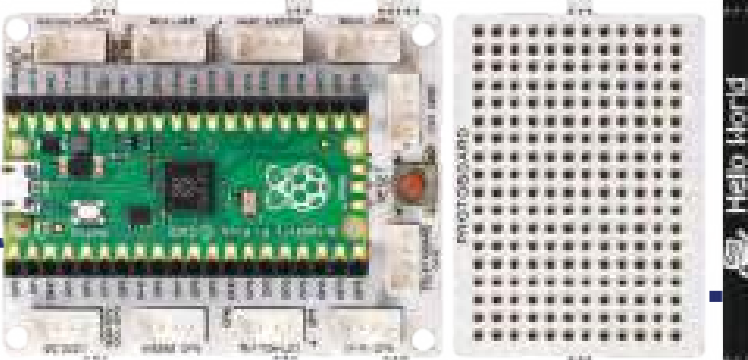
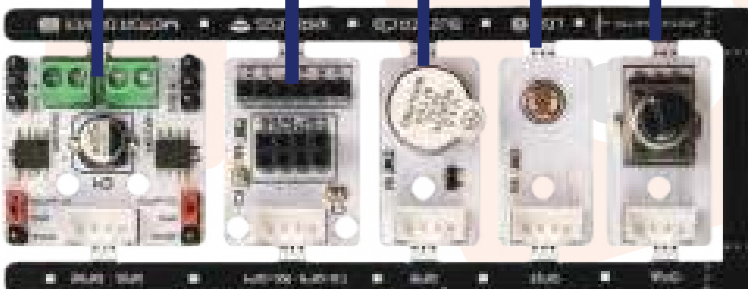
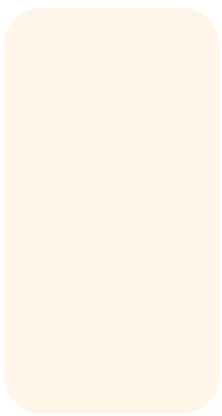
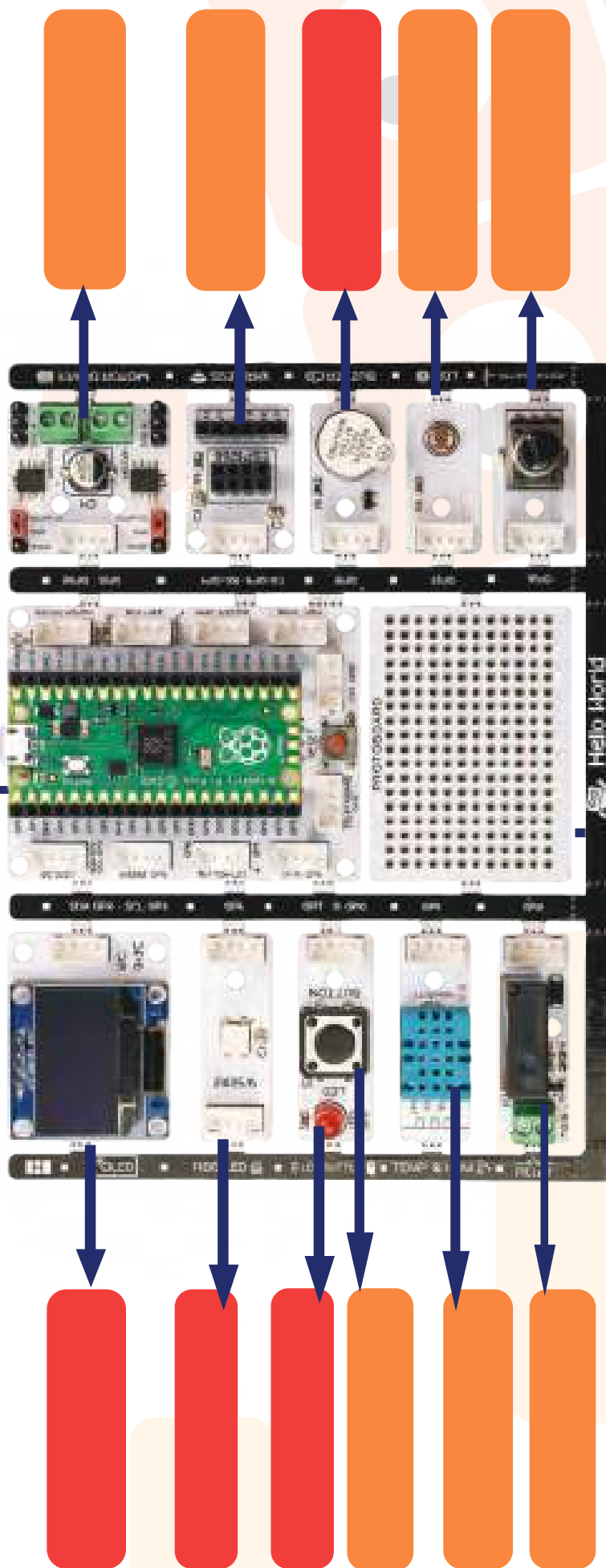
**3. Input Devices:** These are devices that allow the computer to receive information from outside. For example, our ears are one of the input organs of the body, and they send the sounds we hear to our brains. The input elements of the computer are the keyboard, mouse, microphone, touch screen, etc. devices that allow us to send the information received from outside to the computer's processor.

**4. Output Devices:** They are devices through which the computer transmits information. For example, our mouth is one of our organs, we transfer information to the outside with the help of sound by speaking. Computer output devices Monitor, speakers, etc. are devices. The monitor creates the visual outputs and the speaker creates the audio outputs.

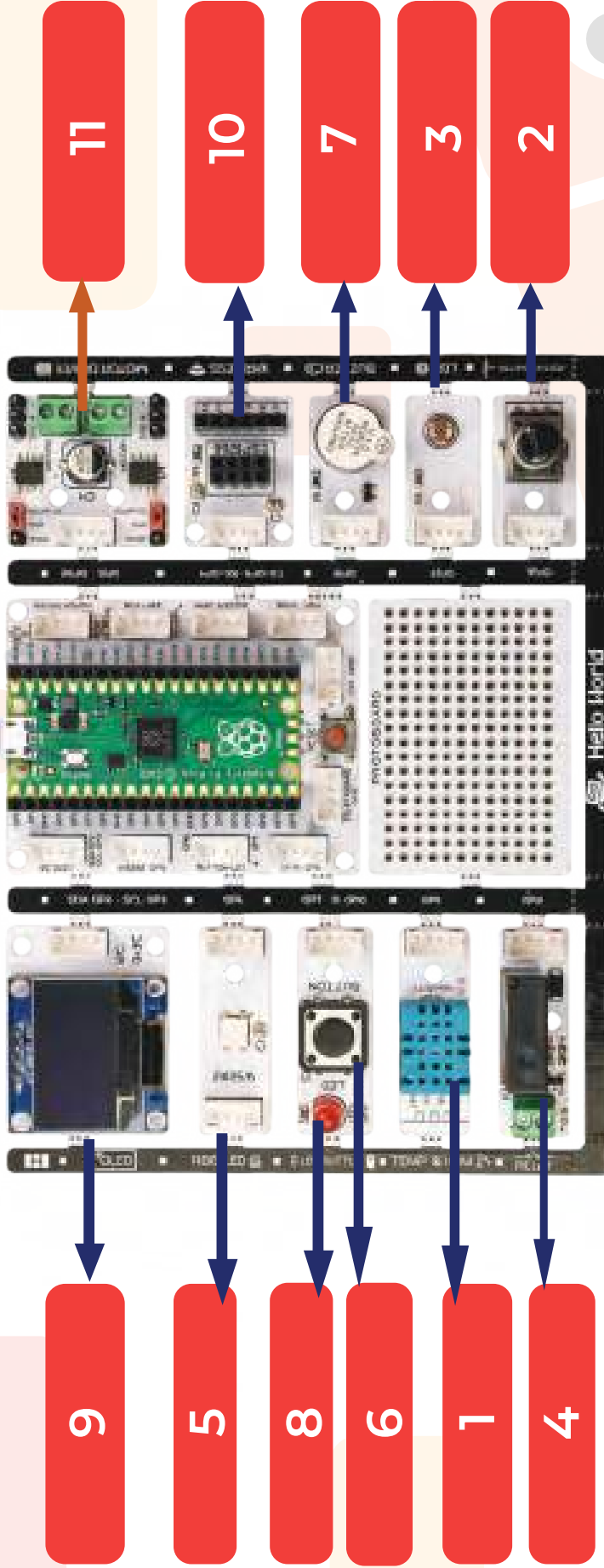
Raspberry Pi Pico



Protoboard



(hint: The largest odd number with different digits)



1. Module with red light output
2. A module that outputs text, images or simple animations
3. Module that outputs sound
4. Module that gives colored light output by mixing red, green, blue colors.



Empty speech bubble

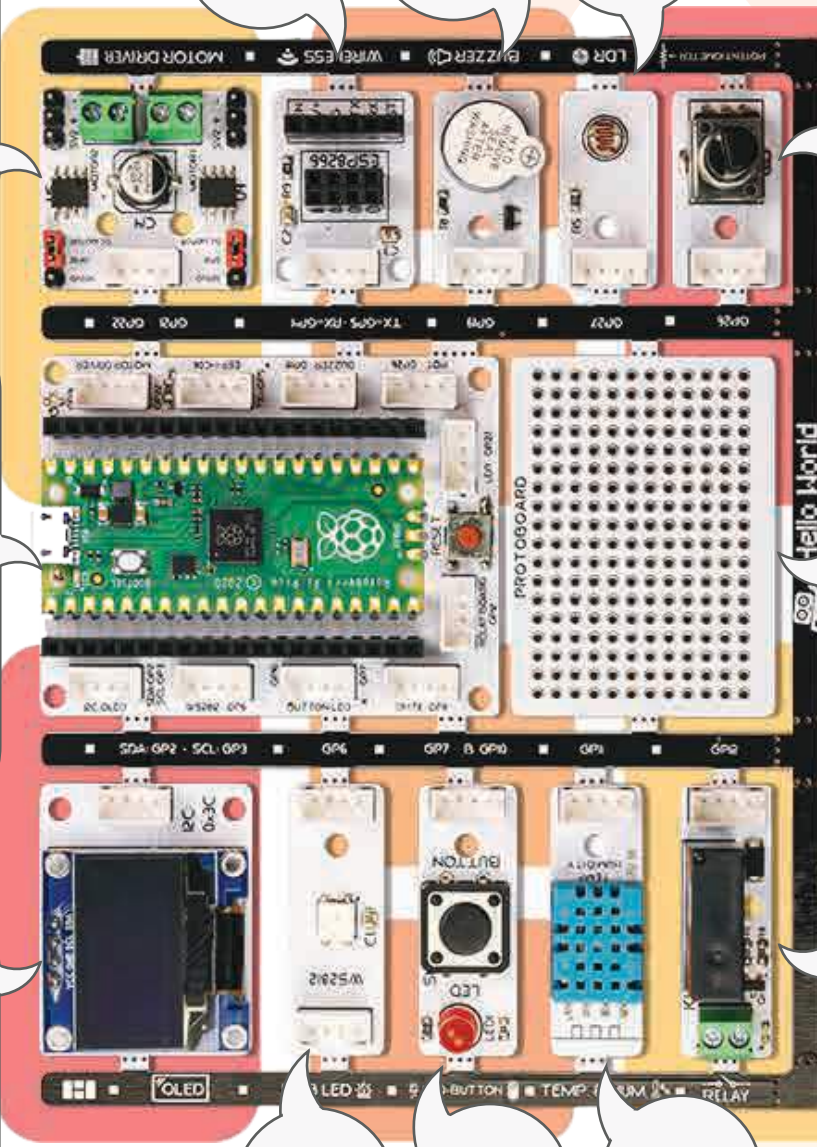
Empty speech bubble

Empty speech bubble

Empty speech bubble

Empty speech bubble

Empty speech bubble



Empty speech bubble

Empty speech bubble

Empty speech bubble

Empty speech bubble

Empty speech bubble

Empty speech bubble



**Buzzer:** It is the module that provides sound.



**OLED Screen:** It is the module where we can get textual, visual or animation type outputs.



**LED & BUTTON:** The red LED module is the circuit element that allows us to output only red light. A button is an input element that we understand whether it is pressed or not.



**RGB (Red-Green-Blue) LED:** It is a circuit element that we can output many colors by mixing Red-Green-Blue colors in certain proportions.



**DHT Temperature and Humidity Sensor:** It is a module that measures the temperature and humidity of the environment.



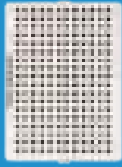
**Relay Module:** It is a circuit element that works when current flows through it. You can think of it like a switch that turns on your lamp.



**LDR Sensor:** It is the module that changes the resistance value according to the amount of light falling on it and thus detects the amount of light in the environment.



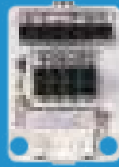
**Raspberry Pi Pico Module:** You use me to code your projects, and you also make the pin connections of the modules and sensors you use from the pins on me. To make an analogy, I am the brain of the PicoBricks.



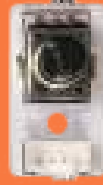
**ProtoBoard:** It is the module that enables any sensor or conductive material not on the card to communicate with the card through the conductive paths and jumper cables on the PicoBricks card.



**Motor Driver:** It is the circuit element that adjusts the speed and frequency of the circuit elements like the motor etc.



**Wireless Module:** It is the module that provides internet access by connecting to the Wi-Fi network in the environment. Thanks to this module, we can communicate with our projects remotely.



**Potentiometer:** It is a resistor whose value can change as you turn the knob.

**Output Blocks:** Code blocks that manage audio and video outputs.

**Pin Blocks:** Code blocks that we use when we connect to PicoBricks pins.

**Operator Blocks:** Code blocks with mathematical operations.

**Data Blocks:** Code blocks that allow us to use the data we have defined for the project.

**Input Blocks:** Code blocks that manage input devices.

**Control Blocks:** Code blocks where we can control run, end, condition, loop etc.

**Variable Blocks:** Code blocks where defining a variable, assigning a value to a variable etc. are made.

**My Blocks:** Code blocks where we can prepare our own blocks and define tasks.



**Libraries:** The section where we can access the code blocks after loading the libraries into the MicroBlocks IDE editor.

when PicoBricks button

initialize I2C OLED\_0.96m address(hex) 5C reset pin# 0 flip

play note A octave 0 for 1000 ms

write A at x 0 y 0 inverse

PicoBricks set RGB LED color

play note E octave 0 for 500 ms

write E at x 0 y 0 inverse

PicoBricks set RGB LED color

play note E octave 0 for 500 ms

PicoBricks set RGB LED color

play note E octave 0 for 500 ms

PicoBricks set RGB LED color

play note E octave 0 for 500 ms

PicoBricks set RGB LED color

play note E octave 0 for 500 ms

PicoBricks set RGB LED color

play note D octave 0 for 500 ms

write D at x 0 y 0 inverse

PicoBricks set RGB LED color

play note F octave 0 for 500 ms

write F at x 0 y 0 inverse

PicoBricks set RGB LED color

play note E octave 0 for 1000 ms

write E at x 0 y 0 inverse

PicoBricks set RGB LED color

play note F octave 0 for 500 ms

write F at x 0 y 0 inverse

PicoBricks set RGB LED color

play note E octave 0 for 500 ms

write E at x 0 y 0 inverse

PicoBricks set RGB LED color

play note F octave 0 for 500 ms

write F at x 0 y 0 inverse

PicoBricks set RGB LED color

After pressing the button, we need to drag these two code blocks to our project page in order to run the screen and other code blocks.

These are the code blocks that allow the tones required for the melody to be played from the buzzer, the tone value to be written on the screen and the color output from the RGB LED module. We can complete our project by copying these blocks and changing the values in them.



# 1) Worksheet (The Loops)

1. When we run the code blocks on the side, we want the numbers between 0 and 5 to be written on the screen. To achieve this, fill in the blank.



2. Fill in the blanks for the MicroBlocks code, which increases the numbers that between 0 and 10 two by two.



3. In the MicroBlocks IDE editor, which of the following options are correct, respectively, the first and last digits that appear on the screen after writing the code blocks on the side?



- A) 8 - 2    B) 8 - 0    C) 10 - 0    D) 4 - 2

## 2. Worksheet

1. Two separate code blocks are given below. In which of the options below are these code blocks correctly given in the order of the last numbers written on the OLED display?



A) 5-1

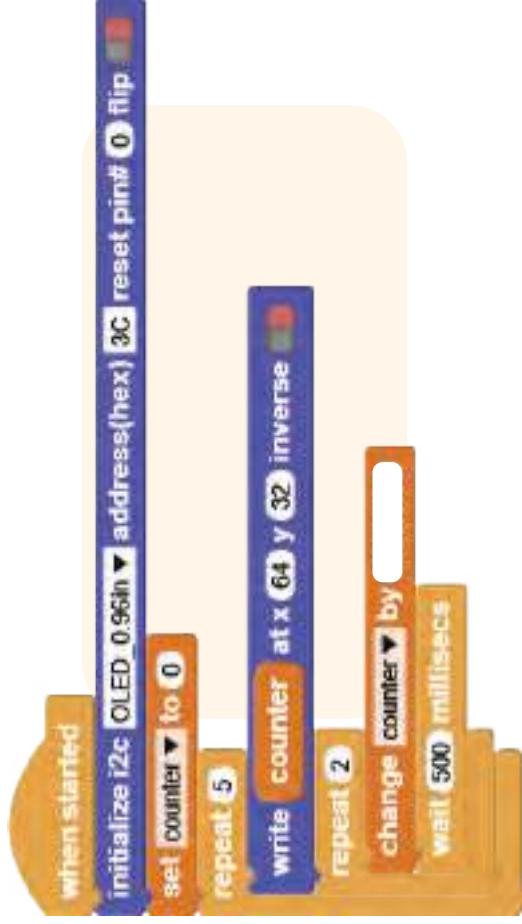
B) 2-5

C) 5-10

D) 10-5



2. Which number comes in the space given in the code block below, the numbers between 0 - 10 (not including 10) are written on the OLED display increasing two by two. Fill in the blank.



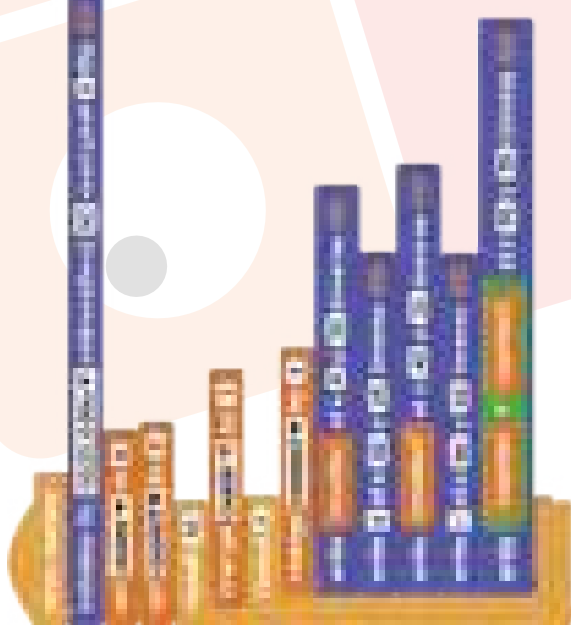
3. Which of the following is the expression written on the OLED display when the code given below is run?

A)  $10 \times 2 = 20$

B)  $2 \times 20 = 40$

C)  $20 \times 10 = 200$

D)  $10 \times 20 = 200$



- Solve the questions on the worksheet below.

	1	2	3	4	5	6
A						
B						
C						
D						
E						
F						

According to the adjacent table, which of the following coordinate points is not colored orange. Indicate by coloring the circle given next to it.

- A5**
- B3**
- C3**
- B4**
- D4**
- A2**
- D2**
- C4**

	1	2	3	4	5	6
A						
B						
C						
D						
E						
F						

According to the adjacent table, fill in the blanks at the coordinate points given below.

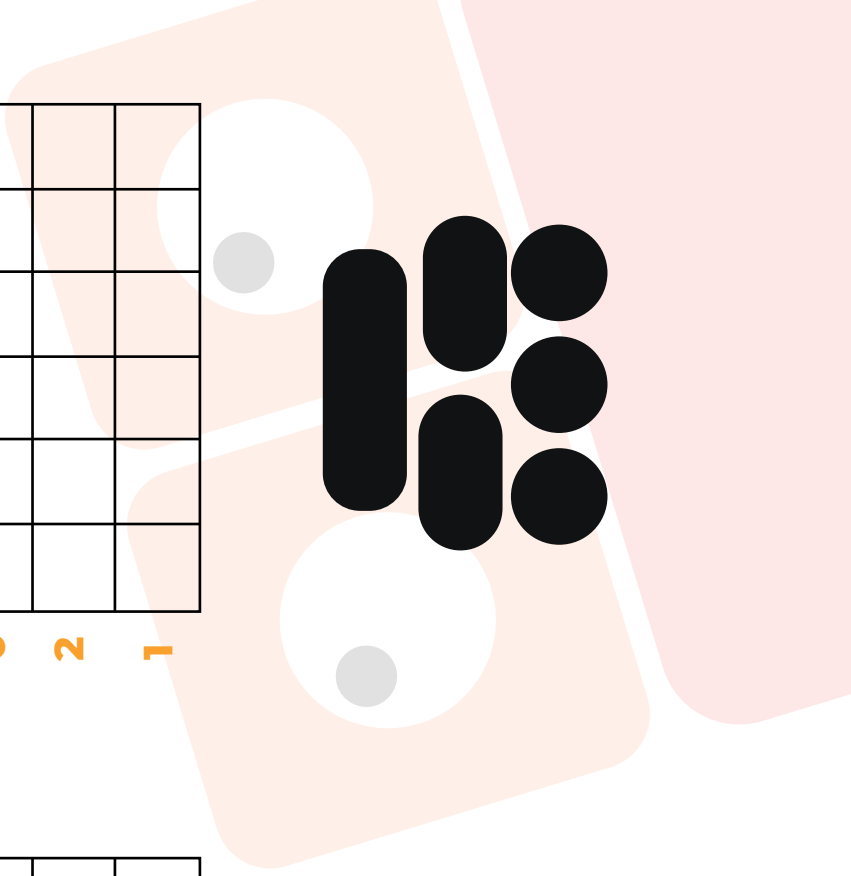
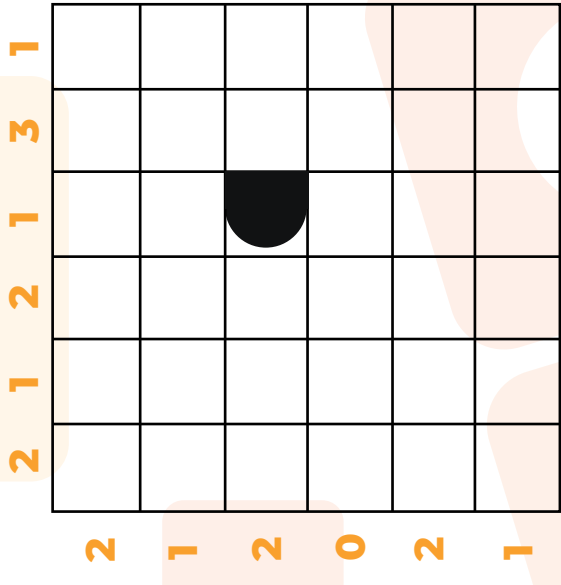
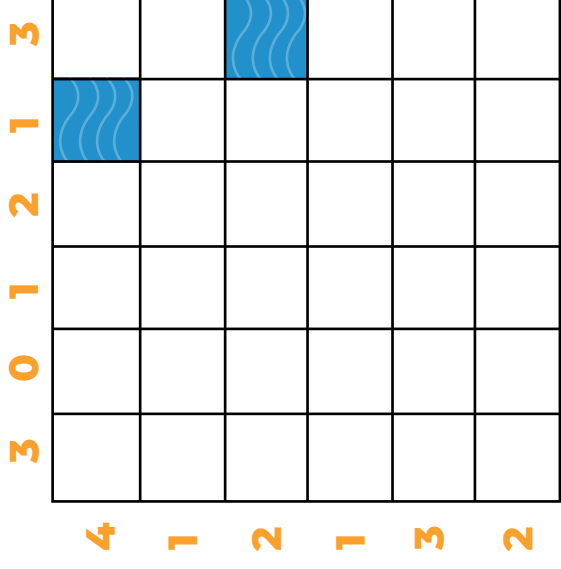
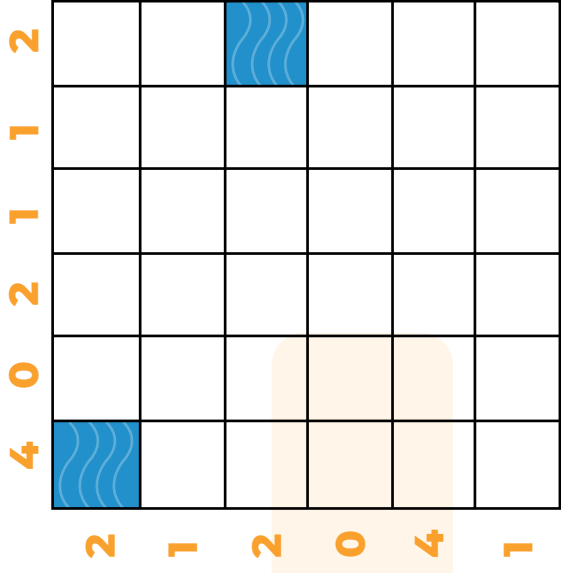
- D\_**
- C\_**
- C5**
- B\_**
- E\_**
- D3**
- E3**
- F\_**

	1	2	3	4	5	6
A						
B						
C						
D						
E						
F						

Indicate the coordinate points given below by coloring the correct places on the adjacent game table.

- A2**
- B2**
- E3**
- E4**
- D2**
- D5**
- A5**
- B5**







# **GALILEO GALILEI**

"We cannot teach people anything; we can only help them discover it within themselves."